October 2014

Infrared Video Tracker: A Study of the Effects of Spatio-temporal Processing on Target Tracking

Benjamin Steven Rude
Worcester Polytechnic Institute

Matthew David Zawatsky
Worcester Polytechnic Institute

INFRARED VIDEO TRACKER

A Major Qualifying Project Report

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

_____

Daniel Banco, PH/ ECE

_____

Benjamin Rude, PH

_____

Matthew Zawatsky, PH

Date: October 14, 2014

Approved By:

_____

Professor Edward A. Clancy, ECE Advisor, WPI

_____

Professor Germano S. Iannacchione, PH Advisor, WPI

_____

Lisa Basile, Affiliate Advisor, Group 108, MIT Lincoln Laboratory

Sarah Curry, Affiliate Advisor, Group 108, MIT Lincoln Laboratory

Emily Fenn, Affiliate Advisor, Group 108, MIT Lincoln Laboratory

Joseph Theriault, Affiliate Advisor, Group 108, MIT Lincoln Laboratory

This work is sponsored by the Department of the Air Force under Air Force Contract #FA8721-10-C-0007. Opinions, interpretations, conclusions, and recommendations are those of the authors and not necessarily endorsed by the United States Government.

SIGNIFICANT PORTIONS OF THIS PROJECT WORK HAVE BEEN REPORTED IN AN INTERNAL LABORATORY REPORT. MOST OF THAT TEXT WILL NOT BE REPEATED HEREIN.

Abstract

Group 108 at MIT Lincoln Laboratory focuses on understanding air defense issues in the United States Air Force. Group 108 extracts the infrared (IR) signatures of targets from IR video captured by IR sensors aboard an airborne platform. This project provided analysts with a tracker to automate the localization of targets in IR imagery. The tracker utilizes thresholding-based target identification and signal processing techniques to attenuate background clutter. Imagery collected by Group 108's Airborne Infrared Imager (AIRI) was used to evaluate the tracker's performance. The tracker's performance was evaluated in terms of the tracker failure rate and pixel position error.

Statement of Authorship

While all team members contributed to the success of this project and acted as main authors and editors, each student was responsible for the main authorship of major sections. All team members edited all sections of the report.

Daniel Banco was the main author of the following sections: Abstract, Sections 1, 2.3.6, 3.1, 3.3.2, 3.3.3, 3.5.2, 4, 5, 5.1, and 5.2.

Benjamin Rude was the main author of the following sections: Sections 2.1, 2.2, 2.4, 3.2, 3.3.4, 3.4.1, 3.5, 5.4, 5.5, Appendix A, and Appendix C.

Matthew Zawatsky was the main author of the following sections: Executive Summary, Sections 2.3, 2.4.3, 2.5, 3.3, 3.4.2, 5.3, 6.0, and Appendix B.

Acknowledgements

We would like to thank our Lincoln Laboratory mentors, Lisa Basile, Sarah Curry, Emily Fenn, and Joseph Theriault, for their continuous support and aid throughout our summer internship and MQP work. We would also like to thank our WPI advisors, Professor Edward A. Clancy and Professor Germano S. Iannacchione for guiding us and providing us with dynamic challenges to overcome.

We would also like to thank Christie Cocanour, Kyle Cochran, Mehul Dhorajia, Antti Koski, John Norstrom, Marianne Procopio, Douglas Reynolds, Eric Rugar, and Joan Spind who all have made significant contributions to assist us in succeeding.

Executive Summary

Group 108 at MIT Lincoln Laboratory focuses on understanding issues with air defense in the United States Air Force. One of the main methods Group 108 uses to diagnose issues is to analyze data collected from sensors, including imagery captured from infrared (IR) sensors. IR light is emitted at high intensities for objects with high temperatures and low intensities for objects with low temperatures. When an aircraft is flying through the sky, the engine is extremely hot, and therefore it gives off a high intensity IR signature. Other factors, such as drag, also heat up other parts of the aircraft. IR sensors have a limited field of view (FOV), but passively gather information from all objects in their FOV. One of the challenges with gathering IR imagery is that objects such as clouds and buildings, classified as clutter, also emit IR signatures. The purpose of this project was to apply image processing techniques to IR imagery to attenuate the presence of clutter and implement a tracking algorithm to accurately and precisely track aircraft moving through the scene. The tool we created to implement image processing and target tracking, the Video Analysis Tool (VAT), was coded in C++ to be consistent with other toolsets that Group 108's post-mission analysis team uses.

The Airborne Infrared Imager (AIRI), the sensor that gathered the imagery which we used for testing, is one of Group 108's IR sensors aboard the Airborne Seeker Test Bed (ASTB). AIRI is Group 108's truth sensor, and it gathers IR imagery in the mid-wave IR (MWIR) (2-5.3 μm) and the long-wave IR (LWIR) (6-10 μm) spectra. The truth data from AIRI is useful for comparing against data from other IR sensors because of its broad range.

In order to gather data to test the results of our image processing techniques and tracking algorithm, we used recorded imagery from AIRI. Because there were no targets in the imagery, we inserted targets as small clusters of pixels at an intensity value of ten standard deviations over the average IR intensity of the scene. The pointing vector of the IR sensor, determined by the roll of the aircraft and the pitch and yaw of the sensor, was used to determine where the target should be positioned in each frame for it to follow a realistic path through space. The position of the inserted target was recorded as truth data and used to analyze our system.

To start tracking, the tracker requires an image and the target's initial position as input. The tracking algorithm first creates a binary, thresholded, image based on a threshold value. If the pixel's value is greater than the mean of all pixels in the image plus two standard deviations of the pixels in the image, it becomes a "1", otherwise it becomes a "0." Next, the tracker

clusters the thresholded image, and removes isolated "1s." The tracker then searches a 9 x 9 region around the previous track position. Within the region, the tracker locates the nearest cluster to the previous track position. The tracker centroids the nearest cluster and records the resulting centroid's position.

After implementing the tracking algorithm, we processed thirteen sets of data. Each data set consisted of 50 frames. The data sets cover various clutter scenarios. Each data set was processed forwards and backwards in time with the tracking algorithm. Our primary metric was tracker failure rate (TFR). We defined the TFR to be

$$\text{TFR} = \left[\frac{\text{number of frames requiring reinitialization}}{\text{total number of frames}}\right] * 100\%.$$

We determined that there was a trend between processing complexity and clutter intensity.

To expand on the initial findings reported in this paper, there are many extensions that can be applied and tested. There are numerous processing techniques that were not implemented for testing. Another feature to consider is a more sophisticated tracker which can predict the position of a target based on its previous movement. A Kalman filter is one technique that could predict position based on previous movement. While focusing on the implemented techniques and algorithms can improve results, testing these algorithms and techniques on data sets with real targets and analyzing across many more clutter scenarios can make trends more apparent. These trends will allow future work to be directed in appropriate directions. Once the trends are determined, an interesting extension would be to create a tool that determines the most effective processing technique based on the content of a scene.

Acronyms

AIRI: Airborne Infrared Imager

GPS: Global positioning system

GUI: Graphical user interface

HgCdTe: Mercury cadmium telluride

InSb: Indium antimonide

IR: Infrared

LL: Lincoln Laboratory

LWIR: Long-wave infrared

MIT: Massachusetts Institute of Technology

MQP: Major Qualifying Project

MWIR: Mid-wave infrared

NUC: Non-uniformity correction

PMA: Post-Mission Analysis

RANSAC: Random sample consensus

RMSE: Root mean square errors

SIFT: Scale-Invariant Feature Transform

TFR: Tracker failure rate

WPI: Worcester Polytechnic Institute

1 Introduction:

Tactical Defense Systems, Group 108 at MIT Lincoln Laboratory, works on air vehicle survivability for the United States Air Force. To study this subject, the group builds and tests various infrared (IR) sensor systems. One of the goals of the group is to measure IR signatures of aircraft and study the issues related to tracking these targets. An IR sensor images a scene by gathering the radiant intensities of all objects in its field of view. These radiant intensities are characterized by an object's physical properties and its environment, which contribute to each object's IR signature. The target signatures are determined by viewing sequences of IR imagery that have been collected by the sensor, locating the target in each frame, and comparing the target's signal to that of the background. The members of Group 108 have recorded data from many flights and would like to be able to view the imagery, read important associated information, and determine the IR signature of an object in the scene. Analysts require a tool for automating the localization of targets in the archived IR imagery. The primary goal of this project was to create and evaluate an infrared video tracker, our Video Analysis Tool (VAT).

The post-mission analysis (PMA) team within Group 108 creates software tools to support the various efforts of Group 108 analysts. The team's C++ application, PMA, currently provides access to a central database and allows users to launch a variety of analysis tools such as MATLAB and a radar receiver calibration tool. We contributed a robust tracking tool to the PMA suite to enable analysts to efficiently and accurately track targets in IR imagery.

Signal processing used in target tracking typically serves to make the target's signal more distinctive with respect to the background signal. The selection of signal processing techniques depends on both the sensor type and target characteristics; however, linear filtering and statistical detection theory are often fundamental to signal processing used in target tracking applications [Richards, 2014, p. 4]. The techniques implemented in this tracking application have been selected based on their direct applicability and accessibility. In this particular application, the object or target is a high intensity point target with no otherwise distinctive shape features. The target's intensity and two-dimensional position in pixel space varies with time. The primary obstacles faced in tracking the target are the dynamic background, clutter, and camera motion.

The tracker needed to be able to distinguish a single point target from the surrounding background clutter. Methods for target identification exist which consider multiple targets and use an estimator such as those implemented by the tracker developed in MATLAB by Group

109. The tracker implemented in this project has been limited to identify only a single target and does not use an estimator such as an alpha-beta-gamma filter or Kalman filter. The tracking algorithm thresholds an image, clusters the resulting binary image, and then identifies the nearest binary cluster as the target. The scope of this project has been limited to this tracking algorithm that, although previously implemented by Group 108 in MATLAB, is not representative of the state-of-the-art within Group 108.

The tracking tool developed was evaluated as a tool to be used in post-mission analysis. In post-mission analysis, an analyst seeking to identify the location of a target in the IR imagery corrects the tracker if it loses track of the target. Correction involves pausing the tracker and reselecting a correct target. To be regarded as an effective tool, the tracker must minimize the number of times analyst correction is required.

2 Background:

The following section will briefly review the physics of the IR spectrum, describe the IR sensors used by Group 108 to produce IR video, target tracking algorithms, and discuss some previously implemented techniques.

2.1 The Infrared Spectrum

The IR spectrum covers a wide range of wavelengths from 0.77–1000 μm, thus between the visible spectrum and microwave radiation. IR sensors detect radiant intensities within this spectrum by focusing incident infrared light onto a focal plane array that absorbs photons on photoconductors. The absorption of a photon on a photoconductor excites an electron from the valence band to the conduction band, which generates a flow of charge that is converted to a voltage using a read-out integrated circuit. The photons detected by the sensors are emitted by blackbody radiation. Blackbody radiation refers to a spectrum of light emitted by any object; objects at room temperature of approximately 300 K emit peak intensity in the IR spectrum. Blackbody radiation can be characterized by the Planck function, which models the spectral radiance emitted by a blackbody. The Planck function is given by

$$I(\lambda, T) = \frac{2hc^2}{\lambda^5} \frac{1}{e^{\frac{hc}{\lambda kT}} - 1},$$

where $I(\lambda, T)$ is the spectral radiance [$W \cdot sr^{-1} \cdot m^{-3}$], h is the Planck constant [J·s], c is the speed of light in a vacuum [m/s], k is the Boltzmann constant [J/K], λ is the wavelength [m], and T is the absolute temperature [K] [Young, 2008]. A graph of the spectral radiance at a room temperature of 300 K is shown in Figure 1.

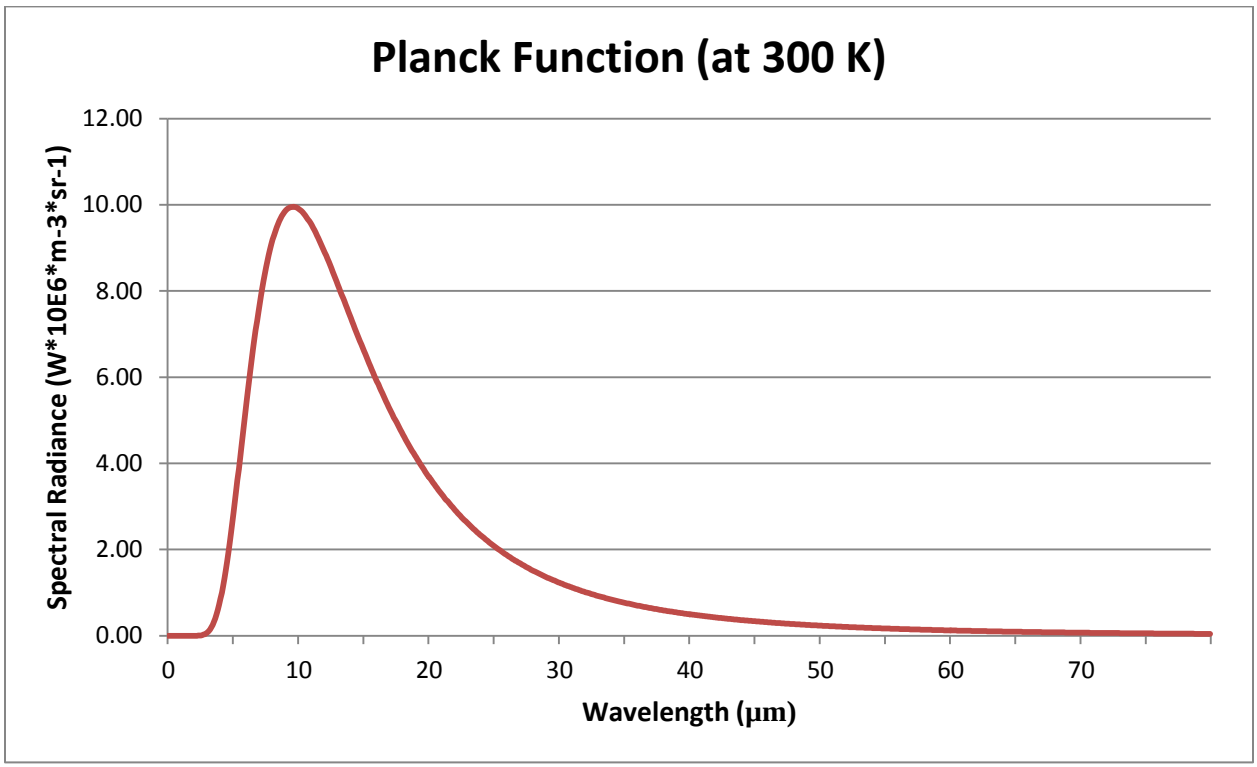

Figure 1: Planck function at 300 K comparing the spectral radiance versus wavelength.

2.1.1 Infrared Sensors

IR sensors use different detector materials including, but not limited to, indium antimonide (InSb) and mercury cadmium telluride (HgCdTe), depending on the application. InSb is a narrow-gap semiconductor, meaning it is a semiconducting material that has a band gap relatively smaller than that of silicon. It is sensitive in the short-wave IR (SWIR) region (1-3 µm) and mid-wave IR (MWIR) region (3-5 µm). HgCdTe has properties that allow for a tunable band gap ranging from the SWIR region to the very long-wave IR (LWIR) region (8-15 µm). Detectors built with HgCdTe work in both the infrared and far-infrared atmospheric windows. However, due to the material's low thermal conductivity, it must be operated cold to be efficient and is usually cooled with liquid nitrogen. These atmospheric windows are bands within the IR spectrum where atmospheric transmission is high and are therefore used for IR detection. The MWIR region corresponds to higher temperatures, peaking at approximately 723 K. This region is better for detecting hot spots in a scene due to its peak temperature. The LWIR has a peak temperature of only 290 K, and therefore is used for lower temperature larger surfaces [Mahulikar et al., 2007]. By imaging a scene in both the MWIR and LWIR spectrum, a sensor can observe targets under different parameters.

2.1.2 Infrared Signatures

The IR signature of an aircraft has multiple contributions including the emission of its engine, the emission due to the mixing of a contrail and the atmosphere, heat caused by friction, and reflections from the sky and earth. The main sources of IR radiation in an aircraft are the engine, nozzle, exhaust, and tailpipe. An aircraft's exhaust causes a plume under certain altitude and humidity conditions, known as a contrail, which contains hot CO_2 and water vapors that cause condensation when mixed into the surrounding atmosphere. As these vapors exit the aircraft at a high temperature, they emit radiation in the IR spectrum. The aircraft skin also emits radiation as it is aerodynamically heated. The other contributions of IR signature of an aircraft are the reflections of skyshine, sunshine, and earthshine. Skyshine is simply radiation in the atmosphere that is reflected off of the airframe. Sunshine is the solar radiation that penetrates the canopy and reflects off of the airframe. Earthshine is the IR radiance that is determined by the temperature and emissivity of the earth's surface. Earthshine depends upon the earth's surface but is usually only important at low altitudes and in the LWIR range. In general, a jet engine aircraft may have IR radiation intensity in the range of 100-1000 W/sr [Mahulikar et al., 2007]. Figure 2 identifies the various contributions to the IR signature of an airborne Falcon-20 aircraft. Group 108 often collects IR imagery of a Falcon-20 during flight testing.

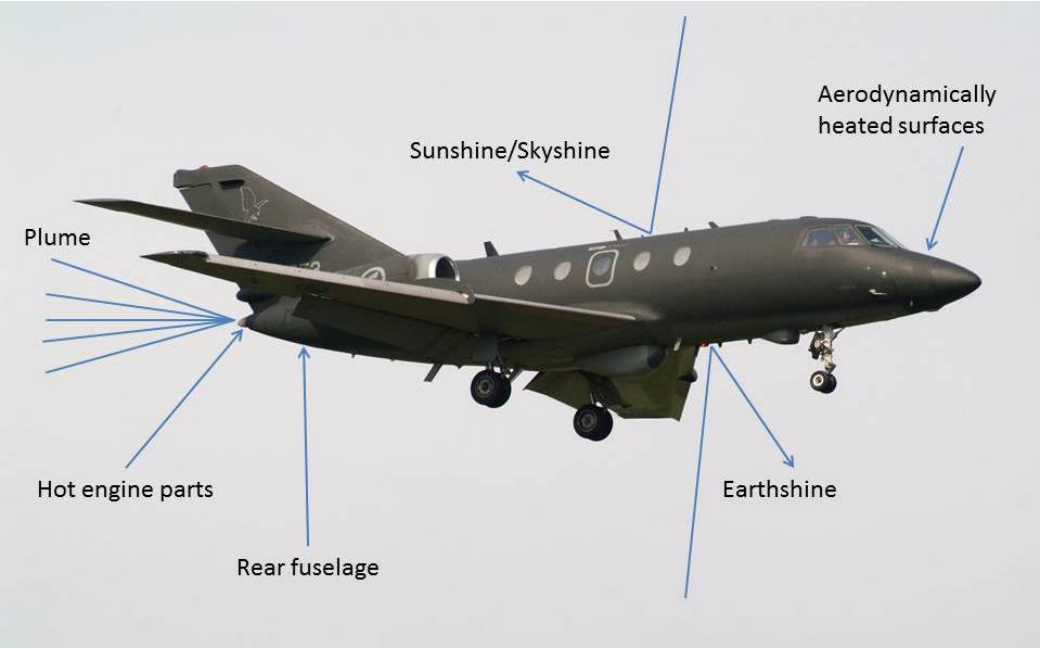

Figure 2: Sources of IR radiance from a Falcon-20 aircraft. Photo taken by [Gunner, 2010].

The viewing aspect of the aircraft also affects the IR signature. When viewing from the rear, the signature is more apparent in the MWIR range because of the high temperature of the engine and exhaust. The aircraft is usually more apparent in the LWIR range if it is being viewed from the front or side because of the lower apparent temperature and larger viewing area.

2.2 Airborne Infrared Imager

The Airborne Infrared Imager (AIRI) is Group 108's calibrated truth sensor that simultaneously collects both MWIR and LWIR imagery using two cooled focal plane arrays that are housed in the same pod. This sensor collects target signatures, evaluates clutter data, investigates the master/slave relationships between radio frequency and infrared sensor functionality, and determines trade-offs for sensor parameters [Klick et al., 1996]. One focal plane array uses an InSb integrated detector-dewar cooled assembly working in the MWIR range with an operating waveband of 2-5.3 μm, seven selectable sub-band filters, and a resolution of 256 x 256 pixels for a 3° x 3° field of view The other focal plane array uses an HgCdTe integrated detector-dewar cooled assembly working in the LWIR range with a waveband of 6-10 μm, five selectable sub-band filters, and a resolution of 128 x 128 pixels for a 1.5° x 1.5° field of view. The AIRI sensor also has a high angular resolution of 200 μrad per pixel for the MWIR focal plane array and a field of regard, or full stable visual field, of ± 45° azimuth and elevation. AIRI uses a shared aperture common telescope with beam splitter design. This catadioptric arrangement offers easier alignment and packing for the sensor even though it has less transmission than an all-reflective system [Klick et al.,1996]. Catadioptric denotes an optical system that reduces aberration, which is the failure of converging rays to focus due to imperfections in a mirror or lens. This type of system reduces aberration by both reflecting and refracting the incident light. The telescope is protected by a multispectral zinc sulfide (ZnS) dome coated with hard carbon and anti-reflection coating. The specifics of the AIRI MWIR and LWIR systems are shown in Table 1. The LWIR focal plane array was replaced in 2010 after imparting a crack. The replacement focal plane array has a resolution of 256 x 256 pixels for a 1.77° x 1.77° field of view. The LWIR imagery used in this project was captured using the replacement focal plane array.

Table 1: AIRI specifications of MWIR, LWIR Amber (1995-2007), and LWIR RVS (2010-present).

	MWIR	LWIR Amber (1995-2007)	LWIR RVS (2010-present)
Focal Plane Array Material	InSb	HgCdTe	HgCdTe
Resolution (pixels)	256 × 256	128 × 128	256 × 256
Data Resolution	12-bit (2^{12} = 4096 counts)	12-bit (2^{12} = 4096 counts)	14-bit (2^{14} =16384 counts)
Field of View (FOV)	2.9° × 2.9°	1.47° × 1.47°	1.77° × 1.77°
Angular resolution (IFOV)	$(200\ \mu rad)^2$ = $(0.011°)^2$	$(200\ \mu rad)^2$ = $(0.011°)^2$	$(120\ \mu rad)^2$ = $(0.007°)^2$
Spectral Band	2-5.3 μm	6-10 μm	6-10 μm
Number of sub-band filters	7	5	5
Operating temperature (Stirling coolers) (K)	77	65	70
Focal length (cm)	19.0	25.0	25.0
Aperture (cm)	11.43	11.25	11.25
f/#	1.66	2.22	2.22
Well Storage Capacity ($x10^7$ electrons)	1.2	4	2.3
Pixel Size (μm)	38 × 38	50 × 50	30 × 30
Quantum efficiency (%)	70	58	>90 (excl. ND cold filter)
Percentage bad pixels	<1%	6%	2%
NEI (noise equivalent irradiance)– temporal (pW/cm^2 at τ_{int} given)	0.013 (now 0.42) at 0.84 ms	0.20 at 0.084 ms	0.38 at 1.0 ms
NEI – spatial (pW/cm^2 at τ_{int} given)	0.014 (now 0.06) at 0.084 ms	0.31 at 0.084 ms	0.07 at 1.0 ms
NEP – temporal ($pW/Hz^{1/2}$)	0.037 (now 1.26)	0.18	1.19
NEP – spatial ($pW/Hz^{1/2}$)	0.040 (now 0.17)	0.28	0.22
NEΔT (noise equivalent temperature difference)– temporal (mK)	144 (now 1443)	302	1121
NEΔT – spatial (mK)	156 (now 209)	467	206
D* (normalized detectivity)– temporal ($cm\ Hz^{1/2}$ / W)	1.02×10^{11} (now 3.0×10^{9})	2.78×10^{10}	2.5×10^{9}
D* – spatial ($cm\ Hz^{1/2}$ / W)	9.49×10^{10} (now 2.2×10^{9})	1.80×10^{10}	1.4×10^{10}

Computers within the aircraft's cabin coordinate the sensor by feeding data to recorders and combining the imagery with information of the aircraft position, orientation, and time using GPS, inertial navigation system (INS), and telemetry sensors that are also onboard [Klick et al., 1996]. AIRI performs in-flight calibration using an extended-area blackbody of 120 cm that resides within the pod housing. The imagers and light-collecting optics are on a gimbal that

rotates 180° towards the blackbody, which is placed behind a window with the same optical parameters as that of the external dome. The temperature of the internal blackbody is varied from 273-323 K and the sensor images the flat field [Truitt, 2007]. During flight, the calibrator is slewed between these temperatures and re-calibration is performed during the setup for passes or when parameters have been changed. A calibration is performed to evaluate each pixel's response to the temperature change. First, pixels with no temperature dependence or those overly sensitive are identified and denoted as bad pixels. Then, the good pixels are evaluated and a gain and offset for each pixel are applied so that the output for each pixel, measured in counts, is on the same scale. The bad and good pixel information is used to create a non-uniformity correction (NUC) file that is applied in both real-time and post-processing.

2.2.1 Group 108 IR Signature Calculation

Once a NUC has been applied to an image, the analyst can then attempt to track a target across time in sequential images to calculate its IR signature. There is a mathematical relationship between a target's signal-to-noise ratio (SNR), signature, size, and range. Group 108 is interested in measuring target IR signatures because analysts are interested in calculating maximum detectable range, which can be found using

$$\mathrm{SNR} = 100\frac{\mathrm{J}\cdot\mathrm{t}-\mathrm{L}\cdot\mathrm{A}}{\mathrm{NEI}\cdot\mathrm{R}^2},$$

where J represents the target signature [W/sr], t is the transmission through the atmosphere [dimensionless], L is the background radiance from target altitude to space [W/sr/cm^2], A is the projected area of the target [cm^2], NEI is the noise equivalent irradiance of the sensor [pW/cm^2], and R is the range between the target and the sensor [km].

By multiplying the emissivity of an object by the Planck function and integrating over the wavelength of the sensor, the counts of temperature can be converted to a radiance count. The counts to radiance conversion should be a linear transformation [Truitt, 2007]. The following relationship describes how to calculate the contrast radiance of the pixel containing a target

$$\Delta\mathrm{L} = \frac{\mathrm{J_{con}}}{\Omega\mathrm{R}^2} = \frac{\mathrm{J_{src}}-\mathrm{L_{bkg}A_{proj}}}{\Omega\mathrm{R}^2},$$

where J_{con} and J_{src} are the contrast and source signatures [W/sr], Ω is the detector solid angle [sr], R is the range [km], L_{bkg} is the background radiance [W/sr/cm^2], and A_{proj} is the target projected area [cm^2], which must be less than the pixel projected area ΩR^2. The minimum detectable

irradiance (MDI) can be found using the contrast irradiance H

$$H = \Omega\Delta L = \Omega\left[\int P(\lambda, T_{ground} + \Delta T)d\lambda - \int P(\lambda, T_{ground})d\lambda\right],$$

where Ω is the detector solid angle [sr], ΔL is the pixel contrast radiance [W/sr/cm^2], λ is the wavelength [μm], P(λ, T) is the Planck blackbody function, ΔT is the pixel contrast temperature [K], and the integrals are evaluated over AIRI wavebands of 2-5.3 μm (MWIR) or 6-10 μm (LWIR) [Klick et al., 2001].

Imagery captured by infrared sensors can have numerous objects cluttering the scene. Image processing techniques can be applied to remove clutter and make non-clutter objects more apparent. For this project, the goal was to track a target within the scene to allow analysts to calculate the target's IR signature efficiently with minimal human intervention.

2.3 Overview of Image Processing

The IR emissions from other objects within a scene such as clouds or ground can clutter an image due to the spatial variation in temperature of the scene, making it difficult for a tracker to extract a target. One way to improve a tracker's performance is by applying image processing to an image to declutter the scene.

2.3.1 Feature Detection

Feature detection algorithms recognize similar shapes between images. One technique for recognizing shapes is called the Scale-Invariant Feature Transform (SIFT). The SIFT algorithm searches the image for keypoints, points that are invariant to scale and orientation. In order to determine the keypoints, SIFT generates a Gaussian scale space. Scale spaces are images seen from different perspectives. For example, when observing a beach, one scale space corresponds to looking at a single grain of sand and another scale space corresponds to observing the whole beach from a helicopter. The mathematical representation of the Gaussian scale space is given by

$$L(x, y, \sigma) = \frac{1}{2\pi\sigma} e^{\frac{-(x^2+y^2)}{2\sigma^2}} * I(x, y),$$

where L is the scale space, x and y are the position in pixel space, σ is the standard deviation, $*$ is the convolution operator, and I is the input image. Convolving the image with a Gaussian produces a Gaussian blur, which effectively reduces the image resolution. Next, in order to find

stable keypoints, points that are invariant across many possible scales, difference of Gaussian scale space terms are introduced which are defined as

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma),$$

where k is a constant denoting the scale separation. The goal is to find keypoints at some point (x, y) that are keypoints for many different k values. The effect of viewing the differences between the scale spaces is comparable to gradually reducing the image resolution [Lowe, 2004]. The Difference of Gaussian is computed over a number of scale space sizes as well as on multiple steps within each scale space. The scale spaces used in the SIFT technique decrease in size by a factor of two for each octave, or successive scale space. Within each of the octaves, there are a number of scales which are images of the same size with reduced resolution. Within each scale, the resolution is roughly cut in half. Depending on the size and resolution of the original image, the number of octaves and number of scales per octave vary. After extensive testing, the creator of SIFT determined that four octaves and five scales per octave were optimal for most image sizes. Four computations of the Difference of Gaussian for five scales of two octaves are shown in Figure 3 [Lowe, 2004].

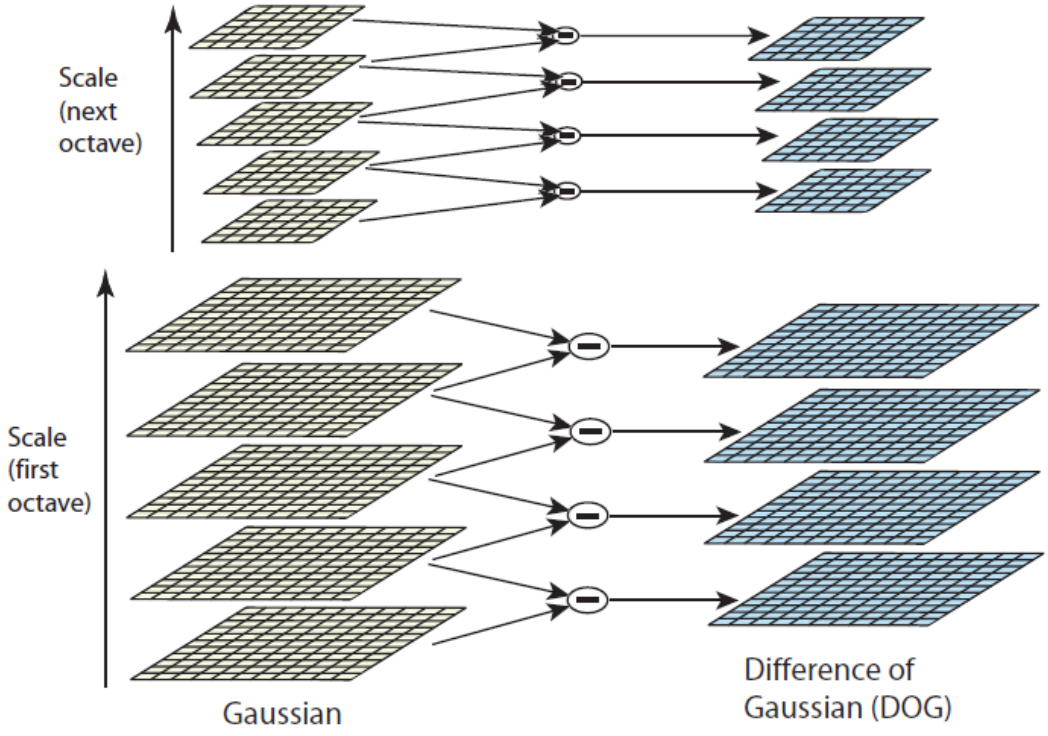

Figure 3: Visualization of octaves, scales, and Difference of Gaussians. (Taken from [Lowe 2004]).

The difference of Gaussian term approximates the Laplacian of Gaussian

$$\sigma^2 \nabla^2 G,$$

where σ is the standard deviation, ∇^2 is the Laplace operator, and G is the Gaussian. The maxima and minima of the Laplacian of Gaussian produce the most stable image features compared to many other techniques. SIFT then determines the positions of the scale space extrema by checking the neighbors of each point. If the point is determined to be an extrema, it is deemed a keypoint [Lowe, 2004]. The red circle in Figure 4 is the point checked as an extrema, and the blue circles are all of the nearest neighbors in scale space. For example, if the red circle in Figure 4 had a value larger than all of the blue circles, its location would be recorded as a keypoint. In certain cases, some of these keypoints are removed due to factors that can cause instability, such as being close to the edge of the image, or being too close to another keypoint.

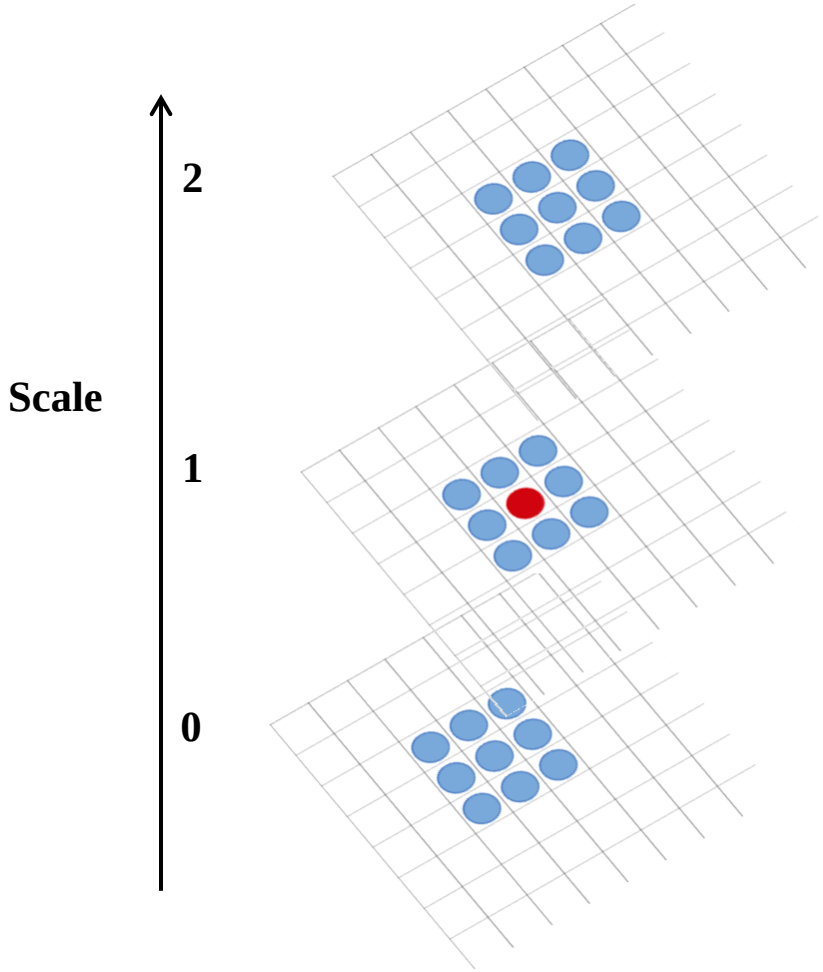

Figure 4: Extrema check method for keypoints. The blue circles are the nearest neighbors of the red circle in scale space.

The next step in the SIFT algorithm is to model each keypoint to fit the scale. The scale of keypoints is modelled using the location and the Taylor expansion up to the quadratic term of the scale space function. Modeling the scale eliminates keypoints found along the edge of the image [Lowe, 2004]. The Hessian matrix is an optimization technique used to determine extrema given by

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix},$$

where H is the Hessian and D is the difference of Gaussian mentioned earlier in this section. It is possible to calculate the eigenvalues α and β knowing that

$$tr(H) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$det(H) = D_{xx}D_{yy} = \alpha\beta,$$

$$r = \frac{\beta}{\alpha}.$$

Using the above equations, we find that

$$\frac{tr(H)^2}{det(H)} = \frac{(r+1)^2}{r},$$

where r is the ratio of eigenvalues. This ratio is equivalent to the maximum factor for the keypoint's ratio of principal curvatures. So, if any keypoints have a ratio of principal curvatures greater than r, they are removed.

SIFT next determines a gradient magnitude and an orientation for each remaining keypoint. These values are used to generate keypoint descriptors which are used in image registration. The equations for the gradient magnitude and orientation are as follows

$$m(x,y) = \sqrt{\big(L(x+1,y) - L(x-1,y)\big)^2 + \big(L(x,y+1) - L(x,y-1)\big)^2},$$

$$\theta(x,y) = \tan^{-1}\frac{\big(L(x,y+1) - L(x,y-1)\big)}{\big(L(x+1,y) - L(x-1,y)\big)},$$

where m is the gradient magnitude, θ is the orientation, and L is the scale space given earlier in this section.

If a position, scale, and orientation are found for a keypoint, they are invariant across a local image region. The keypoints can be used to calculate descriptors that are nearly invariant across a changing scene. The gradient magnitude and orientation are used to take a region around the keypoint location and orient it at an invariant angle. Next, the gradient of that region is taken and the resultant directions are weighted for each sample point. Each point receives eight different directions with different values to create a descriptor. These gradients are taken from regions 16 x 16 around each keypoint. The descriptor, shown in Figure 5, is a 4 x 4 array of bins, each containing eight direction vectors.

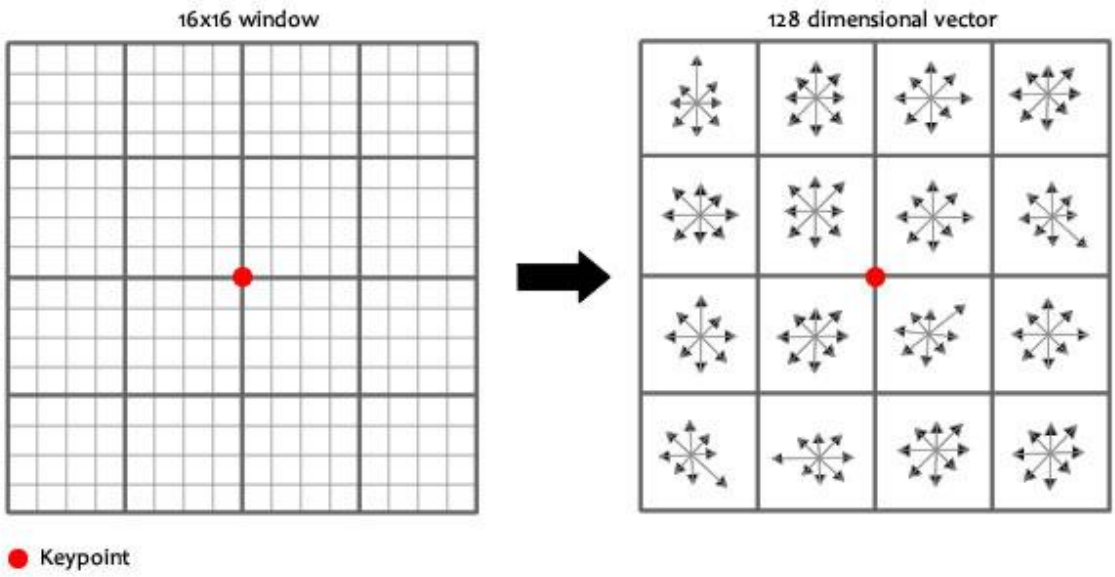

Figure 5: Visual calculation of descriptor from image gradient (Taken from [Sinha, 2010])

The 128 dimensional vector in Figure 5 is the descriptor. Each set of eight vectors is the addition of each part of the gradient in its corresponding sections. Because there are an immense number of possible 128 dimensional vectors, each of these descriptors is unique and can then be used to match keypoints between frames [Lowe, 2004]. After matching the keypoints between frames, it is possible to register the images to move onto later steps in the processing chain.

2.3.2 Keypoint Matching

After a feature detection technique computes keypoints and descriptors, an algorithm must match the keypoints between frames in order for images to be registered. Several techniques exist to match keypoints. The nearest neighbor technique uses one point and finds a corresponding point in the other image by minimizing the Euclidean distance between the two points [Lowe, 2004]. The Best-Bin-First algorithm searches the bins in feature space distally outward from the keypoint's location and finds a matching descriptor [Lowe, 2004].

A simpler alternative to the Best-Bin-First algorithm is the brute force matcher. The brute force matcher iterates through each descriptor in one frame and finds the descriptor in the other frame that has the most similar 128 dimensional vector. For each descriptor the brute force matcher tests, all 128 dimensions are compared to find any variation between the two descriptors

[“Common Interfaces,” 2014]. In some cases, when a scene has little variation, multiple descriptors will match to the same point because the 128 dimensional vectors are not unique.

2.3.3 Spatial Filter

Spatial filters operate in the space domain. The image plane is a spatial domain consisting of discrete pixels. In this domain, spatial filters directly manipulate the pixel values of an image. For linear spatial filters, each pixel is manipulated by multiplying each surrounding pixel by a coefficient defined by the spatial filter kernel and summing the results. An m x n linear spatial filter has a kernel which consists of $m \cdot n$ coefficients [Gonzalez et al., 2004]. The kernel is convolved with an image to produce a spatially filtered image by carrying out the operation shown in Figure 6 for each pixel in the original image. When carrying out this operation for pixels on the border of the image, some coefficients of the kernel lie outside of the domain of the image. It is possible to resolve this issue by adding pixels around the border of the image. The pixels can be given values by mirroring pixels along the borders, by assigning their values to the image mean, or by simply assigning their values to zero. For larger spatial filter kernels, one would need to add more padding to the image to ensure that the kernel does not exceed the image borders.

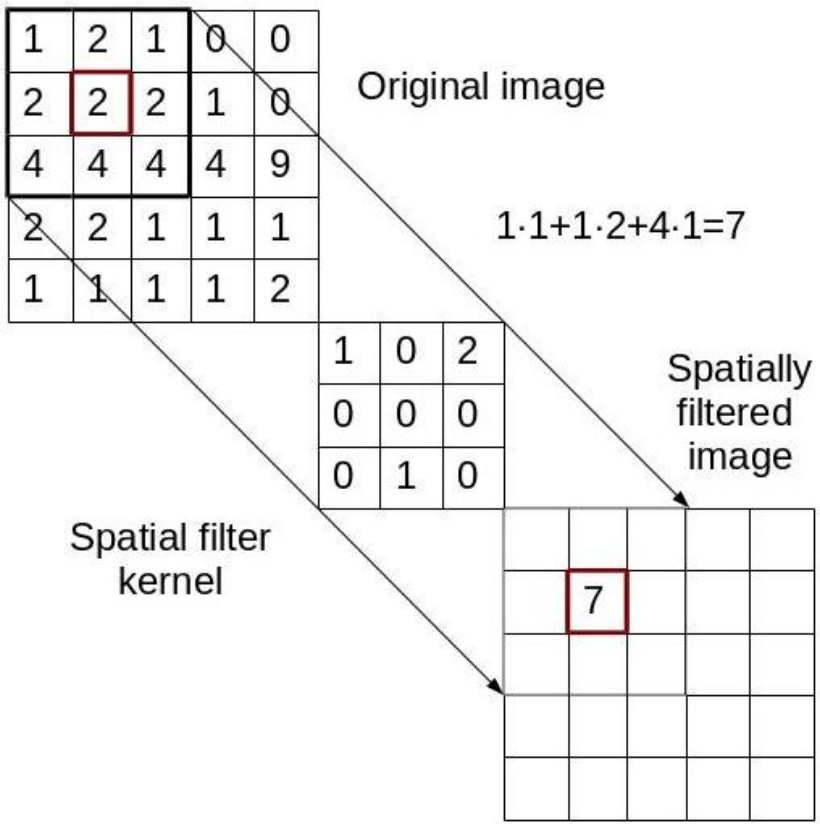

Figure 6: Example of applying a 3 x 3 spatial filter kernel to a 5 x 5 image. Convolving the kernel with this image entails carrying out the operation shown for each pixel of the original image.

An image is represented in the (x, y) coordinate plane in Figure 6 where the origin lies at the top left-hand corner of the image. The spatial frequency in an image along either direction is the frequency with which pixel intensity values modulate [Lehar, n.d.]. The spatial frequency of an image is defined as the number of cycles it makes per radian. The highest spatial frequency features in the image correspond to features with the most detail. In order to understand the effect of applying a linear spatial filter to this kind of imagery, we will look at the values of one row of pixels along the $\hat{\mathbf{x}}$ direction and one column of pixels along the $\hat{\mathbf{y}}$ direction. These pixels are highlighted in white in Figure 7. It is evident from Figure 8 that the pixel values along the column are uniform and span a low spatial frequency spectrum. This column of pixels can be seen to pass through a mostly dark, uniform background in Figure 7. The pixels along the row, however, have more drastically changing values and a frequency spectrum ranging from 0 to 40 cycles per pixel. The row of pixels can be seen to pass over features of varying intensity values such as roofs, houses, roads, and grass.

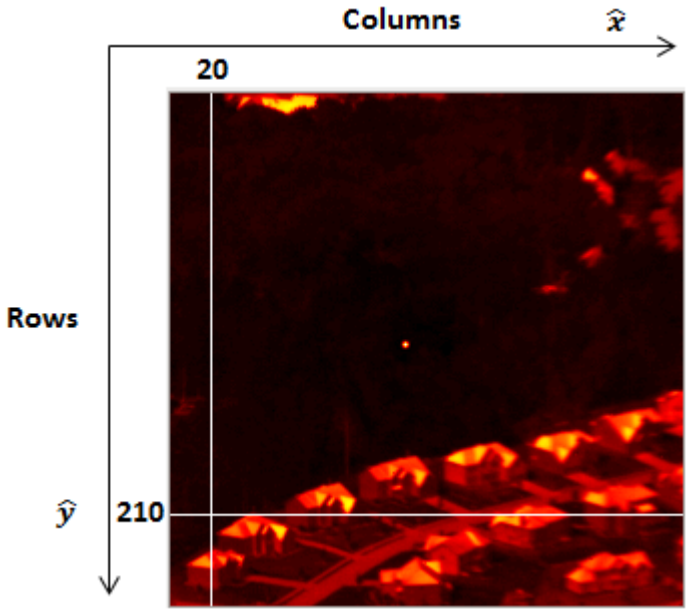

Figure 7: Image of suburban area from AIRI Flight 828. Row 210 and column 20 are highlighted in white.

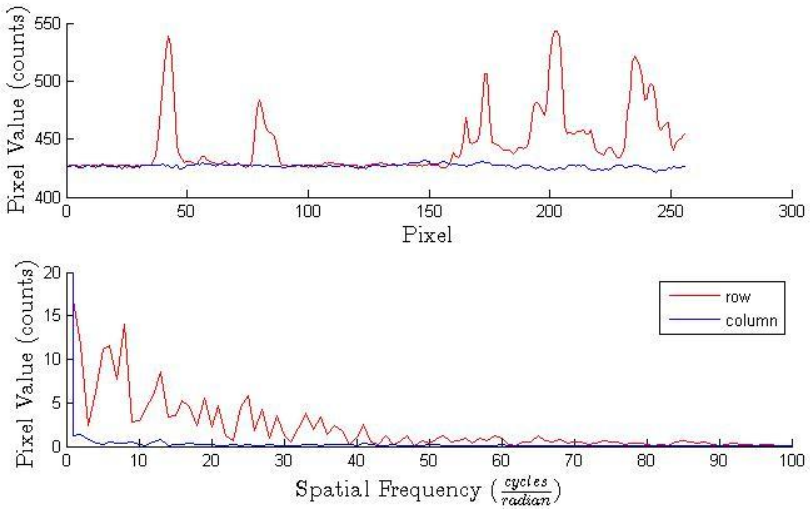

Figure 8: Plot of pixel intensity values along row 210 and along column 20 of frame 12 from AIRI flight 828 (top). Plot of magnitude response along row 210 and column 20 (bottom).

A rotationally invariant 5 x 5 Laplacian spatial filter with guardband can be used to declutter the IR imagery [Klick, et al., 2001]. This linear spatial filter has the kernel

$$h[n_1, n_2] = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & 0 & 0 & -1 \\ -1 & 0 & 16 & 0 & -1 \\ -1 & 0 & 0 & 0 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \frac{1}{16}$$

where the point n_1, n_2 lies at the center. The impulse response of this spatial filter is given by

$$h[n_1, n_2] = \frac{1}{16}\left(16\delta[n_1, n_2] + \sum_{i=n_2-2}^{n_2+2} -\delta[n_1 - 2, i] - \delta[n_1 + 2, i] + \sum_{k=n_1-1}^{n_1+1} -\delta[k, n_2 - 2] - \delta[k, n_2 + 2]\right)$$

Taking the z-transform of $h[n_1, n_2]$ gives

$$H(z_1, z_2) = \frac{1}{16}\left(16 - (z_1^{-2} + z_1^2)\sum_{i=-2}^{2} z_2^i - (z_2^{-2} + z_2^2)\sum_{i=-1}^{1} z_1^i\right)$$

The Fourier transform is then

$$H\left(e^{j\omega_1}, e^{j\omega_2}\right) = \frac{1}{16}\left(16 - (\cos(2\omega_2) + \cos(\omega_2) + \frac{1}{2})\cos(2\omega_1) - (\cos(\omega_1) + \frac{1}{2})\cos(2\omega_2)\right)$$

where ω_1 is the spatial frequency in the x direction, and ω_2 is the spatial frequency in the y direction. The frequency response of this spatial filter is shown in Figure 9.

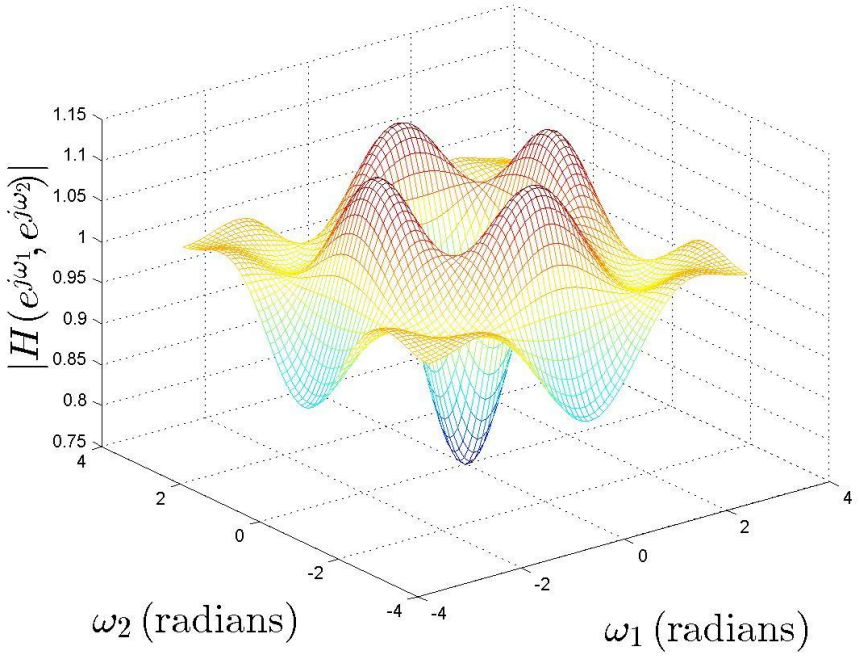

Figure 9: Plot of frequency response magnitude for a rotationally-invariant Laplacian spatial filter with guard band. Domain: $-\pi \le \omega_1, \omega_2 \le \pi$.

This kernel can be represented as the difference between two other kernels.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \frac{1}{16} = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & 0 & 0 & 0 & -1 \\ -1 & 0 & 16 & 0 & -1 \\ -1 & 0 & 0 & 0 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \frac{1}{16}$$

Original — Average of distant pixels — Result

Applying a spatial filter using the left kernel produces the original image; the center kernel produces an image in which each pixel is replaced with the average of sixteen distant pixels; the right kernel, the resulting kernel, produces an image in which each pixel is reduced by the average of sixteen distant pixels. Pixels with similar or lower intensities than distant surrounding pixels will have their intensities reduced to zero or almost entirely to zero. Pixels with higher intensities than distant surrounding pixels will maintain their intensities. As a result, a background with roughly uniform pixel intensity will be attenuated. Figure 10 shows how the spatial filter reduces the intensities of these pixels; roads, roofs, and land of all intensities are black in the spatially filtered image. The images shown are mapped on a hot color scale between their minimum and maximum values. Figure 9 shows that this spatial filter suppresses features with a low spatial frequency. Uniform pixel intensity value regions have a low spatial frequency. Figure 10 shows that pixels with higher intensities than any surrounding pixels remain after applying a spatial filter. Various isolated high intensity points in the temperature image are still present in the spatially filtered image due to the filter's convolution. The target of interest in some infrared imagery is a point target of this isolated high intensity type. Applying this spatial filter will help to isolate the target, however, it will also isolate any point targets present in background clutter as shown Figure 10.

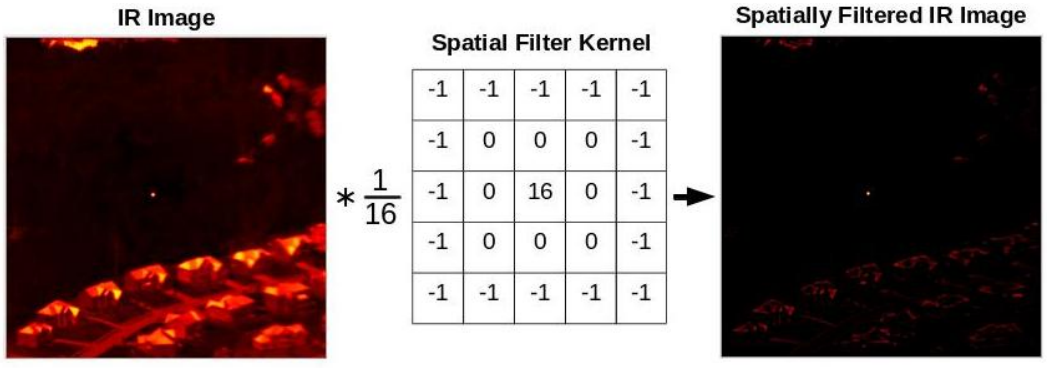

Figure 10: Spatially filtered temperature image from Flight 828. MWIR image of suburban area (left). Rotationally-invariant 5 x 5 Laplacian spatial filter with guardband (center). Spatially filtered image of suburban area (right). An inserted Gaussian point target is apparent in the original image near the center and it persists in the spatially filtered image.

2.4 Target Identification

Within the scenes that IR sensors capture, there is an object, or target, of interest that an analyst would like to track and gather information about. If a target gives off a high intensity IR signature, there exist algorithms within Group 108 to extract the target from the scene and record its signature and location from frame to frame. Oftentimes, there are obstacles that clutter the image with their respective IR signatures inhibiting the tracker's performance. The tracking algorithms must be robust in order to maintain track on targets in high clutter scenes.

2.4.1 Thresholding

A common first step of tracking is to threshold an image to detect possible targets. The tracker calculates a threshold based on the mean and standard deviation of pixel values in a frame or in a region of a frame. Four ways of thresholding an image were explored: static local, static global, adaptive local and adaptive global. Static thresholding involves finding the mean and standard deviation of one image and applying a threshold to every subsequent image with those conditions. Adaptive thresholding is a technique in which each frame has a different threshold level based upon the pixels in that frame. Local thresholding segments an image into blocks and calculates the mean and standard deviation of each block applying a threshold to just the pixels within that block with the parameters of the block. Global thresholding uses the complete image to compute the mean and standard deviation to select a threshold value to be applied to the entire image. The threshold level is set as a certain number of standard deviations above the mean. The number of standard deviations depends on the application and image. For a set of images that change greatly, adaptive thresholding would be more effective due to the changing of parameters with each image. For a highly cluttered image, local thresholding may be more effective by minimizing the effect of clutter in the block surrounding the high intensity target. The threshold value is used to convert an image into a binary image.

2.4.2 Binary Conversion

An image is a set of pixel values, each at a separate coordinate. The threshold value is used to convert this image of pixel values into a binary image with only two colors of pixels, black and white, represented by the numbers "0" and "1" respectively. The binary conversion algorithm scans through the image pixel-by-pixel and if the pixel value is above the threshold level it is rewritten as a "1", otherwise it becomes a "0". An example of a binary conversion is

shown in Figure 11. The binary images clearly show which pixels exceed the threshold value as they are shown in white.

This binary conversion first filters any lower intensity pixels out of the image and accentuates the other values. From here the clustering algorithm is performed to group pixels that are close together and eliminate any random pixels that may have been thresholded.

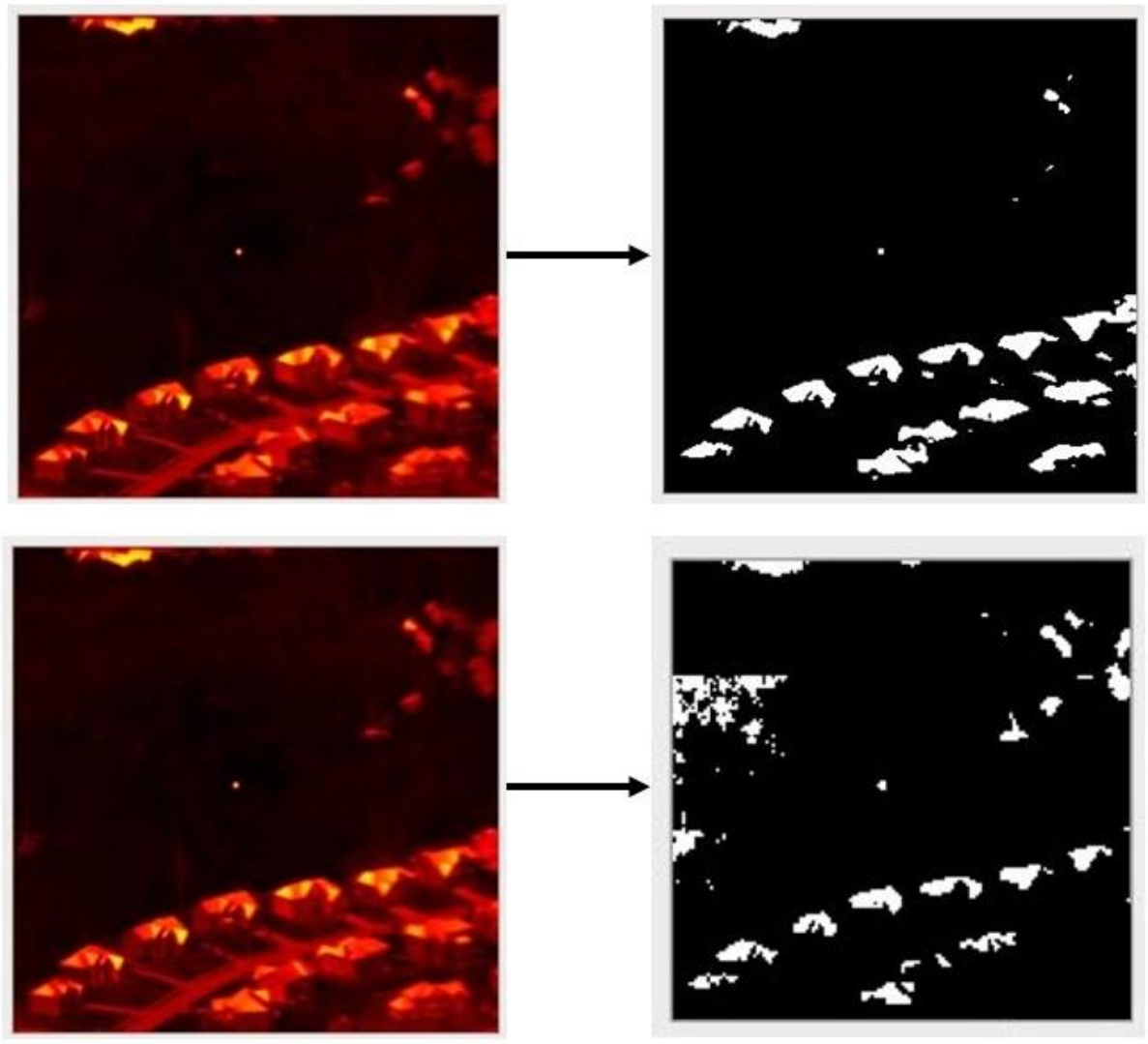

Figure 11: Shown on the left, example MWIR images of data with an inserted target in the center of each frame. In the top, the image has been globally thresholded and converted into a binary image. High intensity pixels greater than the threshold value have been converted to a white value of 1 while all other pixels are black. The target is still apparent in the center. Shown in the bottom, a locally thresholded image with the target still apparent in the center of each frame, but surrounded by false targets.

2.4.3 Clustering

Clustering is the second step in target detection. One of the benefits of clustering is that many possible false targets are eliminated. The result of a clustering algorithm applied to the thresholded image shown in Figure 11, is shown in Figure 12.

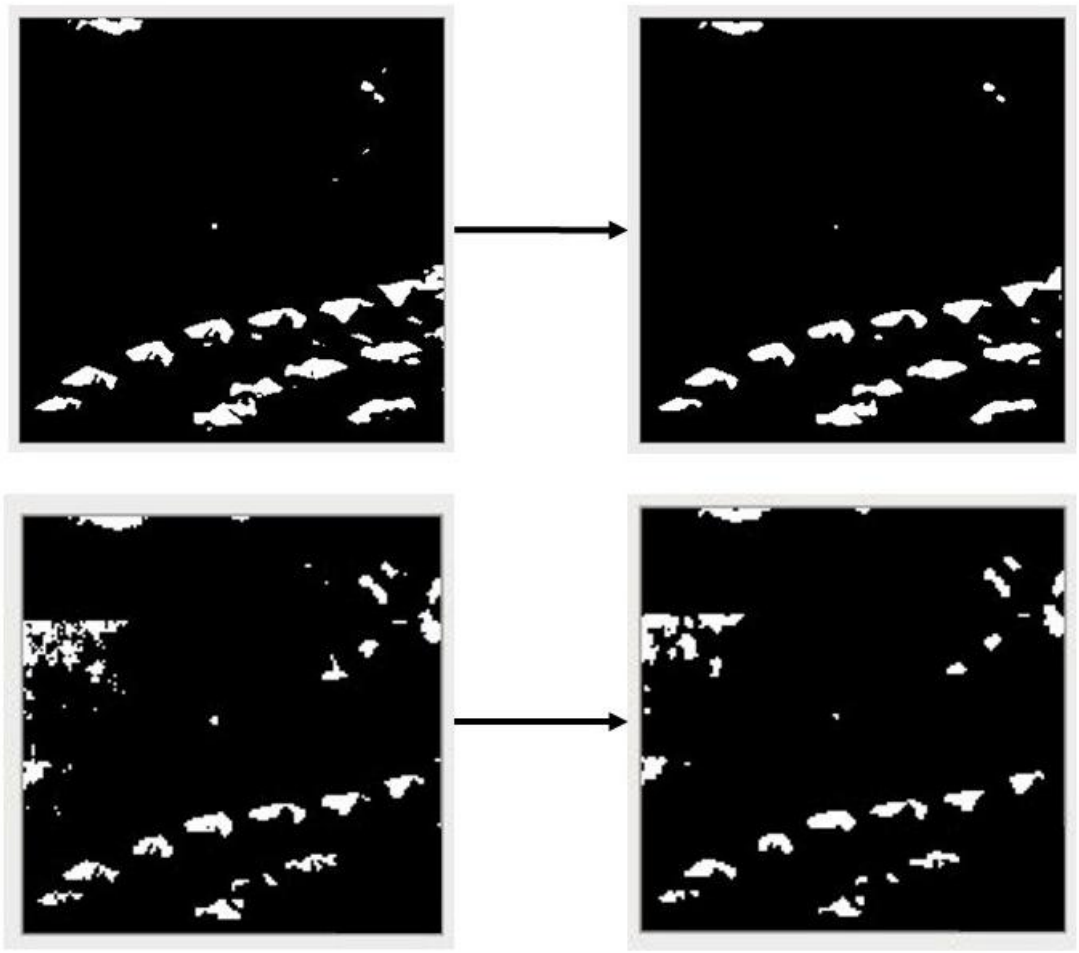

Figure 12: Shown on the left, example images of MWIR binary data after global thresholding (top) and local thresholding (bottom). On the right, the images have been clustered. The target is still apparent in the center.

On the left, there are a number of scattered white pixels which a tracking algorithm could incorrectly identify as a target. After clustering however, most of the scattered pixels are removed, and the resulting shapes are much more defined.

The clustering algorithm passes a matrix of "1's" over each pixel value and determines whether there are a group of "1's" in the area. If this matrix finds a cluster of a certain size, it will set the current pixel to "1"; otherwise it will set it as "0", as shown in Figure 13.

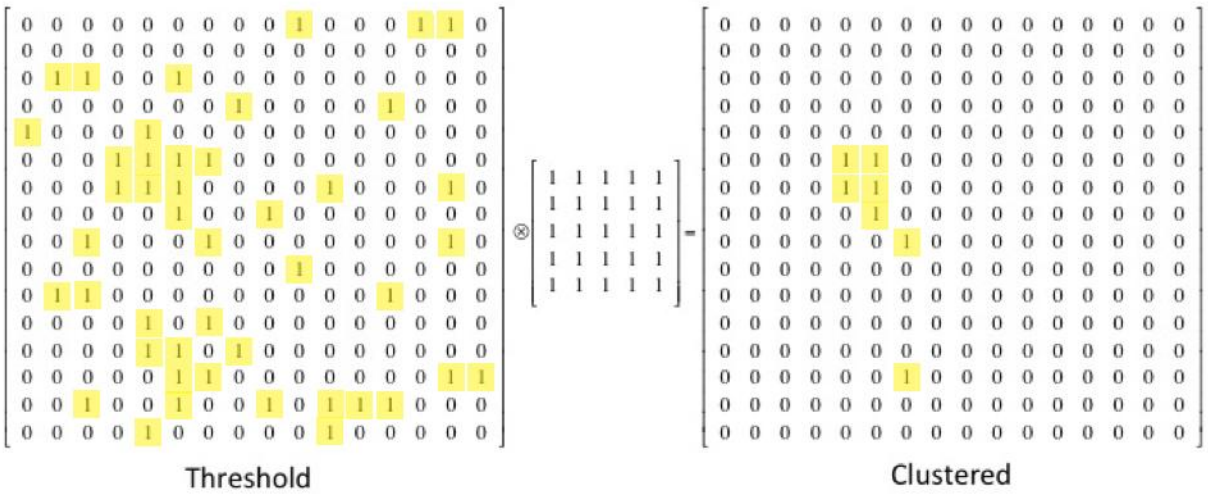

Figure 13: Clustering representation. The matrix on the left is an example of a thresholded image. Applying the clustering algorithm produces the image on the right. The clustering algorithm eliminates isolated "1's" highlighted yellow in the threshold matrix, on the left, while emphasizing large groups of 1s from the threshold matrix.

The clustering algorithm convolves a matrix of "1's" with the thresholded image. If a value in the convolution is greater than a set value, the clustered image's corresponding position is set to a "1". Clustering eliminates any isolated pixels within the image, as shown in Figure 13. From Figure 11 to Figure 12, it is apparent that the clustering algorithm eliminated isolated pixels that were above the threshold. Each cluster became an identifiable feature that the tracking algorithm could then attempt to track.

2.5 Group 108 Capabilities

Group 108 has previously researched the field of target tracking and has created and implemented two algorithms of differing complexity in MATLAB, each for use in tracking single targets during post-processing. There is a basic "blue-sky" tracker that, after receiving an initial target position from a user, loses track of the target 14% of the time in cloud clutter situations and 50% of the time in ground clutter situations [Wagner 2013]. This track failure rate was determined by calculating the percentage of user interventions versus the total number of frames tracked. There is also another tracker, the Group 108 Tracker, which utilizes some additional features that decrease the tracking failure statistics to 1% in cloud clutter and 4% in ground clutter [Wagner 2013].

The blue sky tracker operates under a very simple concept. The first step is for the user to manually select the location of the target in the first frame. The next step is for the algorithm to

find the mean and the standard deviation of the image's pixel values. The function thresholds and converts the image into a binary image. If a pixel's value is less than the mean of the entire frame plus two times the standard deviation, the binary value is a "0" (black pixel); otherwise, the binary value is a "1" (white pixel). This thresholding is described by

$$\text{Pixel value} < mean + 2 \cdot \sigma \rightarrow Pixel\ value = 0,$$

$$\text{Pixel value} \geq \text{mean} + 2 \cdot \sigma \rightarrow \text{Pixel value} = 1.$$

Operating on the binary data, a function clusters all the "1's". The tracker selects the cluster closest to the location of the previously recorded track position. The blue sky tracker executes this algorithm frame by frame and records all track positions and the signature emitted, calculated as described in Section 2.2.1.

A tracking algorithm previously implemented in Group 108 utilizes phase correlation, a Kalman Filter, and a spatial filter. Phase correlation is the equivalent of a translation image registration technique described in Section 1.1.1. The spatial filter is identical to the concept described in Section 2.3.3. Figure 14 outlines the steps taken in the Group 108 tracker.

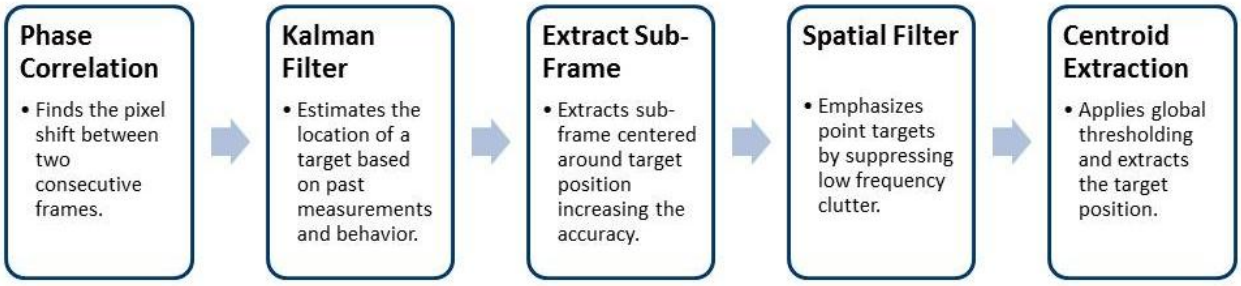

Figure 14: Steps of the Group 108 Tracker.

Phase correlation correlates frames using the frequency domain to find a pixel shift between consecutive frames. In this tracker, phase correlation serves to improve the estimate of the Kalman filter by computing the pixel shift between subsequent frames. The Kalman filter then uses the output of the phase correlation and evaluates the past behavior of the target to estimate the new location in the next frame. The tracker then extracts a sub-frame around the Kalman-estimated target position and applies a spatial filter to localize the target, increase accuracy, and suppress low frequency clutter. The final step then applies a global thresholding technique to the sub-frame and extracts the location of the centroided target.

3 Methodology:

This chapter details the methods used to track moving targets in IR imagery. We preview the data used and explain how targets are inserted into data sets without targets. Then we discuss the overall tracker as well as the techniques used within. Next, we offer a rationale for the parameters chosen throughout the process. Finally, we describe the metrics that quantify our results and how we gathered those data.

3.1 Introduction

The goal of this project was to effectively track single targets from IR sensor data in PMA using the VAT we created. The metrics of interest are those that provide a means for evaluating the success of the tracker for the selected IR imagery. We used the tracker failure rate (TFR) and root mean square error (RMSE). Although the algorithms are not necessarily state-of-the-art, it is of interest to use them to implement a tracker, study their efficacy for this application, and integrate them into Group 108's toolset.

3.2 Data Acquisition

Group 108 has many data sets recorded from flight tests that have been conducted since the mid-1990s. In order to track targets and compare the performance of our tracking and processing algorithms, we selected multiple data sets with various clutter scenarios. Unfortunately, not all of the data sets to which we have access contain targets. To remedy this problem, we characterized the behavior of targets in the imagery that contains targets and have replicated them by inserting simulated targets into our data sets.

3.2.1 Background Clutter Scenarios

We determined the clutter scenarios that are typical and useful for our study. Within the many data sets available, we selected sets with background scenes ranging from low to medium to high clutter scenarios. These data sets have been classified based on the variation of imagery within the scene rather than by the intensity values. We selected 13 total cluttered data sets, each set consisting of 50 sequential frames of both MWIR and LWIR data, shown in Table 2. Images

of each data set can be found in Appendix A. When conducting tests, the 50 frame data sets were played forward and backwards in time.

Table 2: Cluttered data set flight numbers, frame numbers, names, clutter levels, and descriptions.

Flight Number	Frame Numbers	Clutter Name	Clutter Level	Description
826	8025-8074	Ocean	Low	Moderate intensity ocean background containing waves
828	3465-3514	Forest 2	Low	Moderate intensity forest background containing sparse trees and grassy lands
886	5195-5244	Flat Clouds 1	Low	Low intensity flat cloud-bed with blue-sky background
886	5670-5719	Flat Clouds 2	Low	Low intensity flat cloud-bed with blue-sky background
845	6785-6834	Segmented	Medium	Segmented scene with clouds visible at the top third, a distant city in the middle third, and coastline in the bottom third
826	7850-7899	Coastline	Medium	Coastline images containing high intensity ocean clutter versus low intensity coast
828	4265-4314	Runway	Medium	High intensity airport runway visible within a low intensity forest background
826	8861-8910	Forest 1	Medium	Low intensity forest background with few high intensity artifacts between trees
826	8140-8189	Farmland	High	Low intensity forest surrounding high intensity farmlands
826	8765-8805	City	High	High intensity city background containing houses, skyscrapers, and other buildings
828	3657-3706	Suburbs	High	High intensity houses and roads in a suburban neighborhood are visible with low intensity forest surroundings
838	3950-3999	Airport	High	High intensity buildings, roads, and runways with low intensity grassy areas located throughout
868	11912-11961	Clouds	High	High intensity clouds with a blue-sky background

An example of the low clutter scenario is the ocean data set taken from Flight 826 shown in Figure 15. The ocean background contains a somewhat consistent background with periodic waves. An inserted target can be seen toward the bottom left of both the MWIR and LWIR frames.

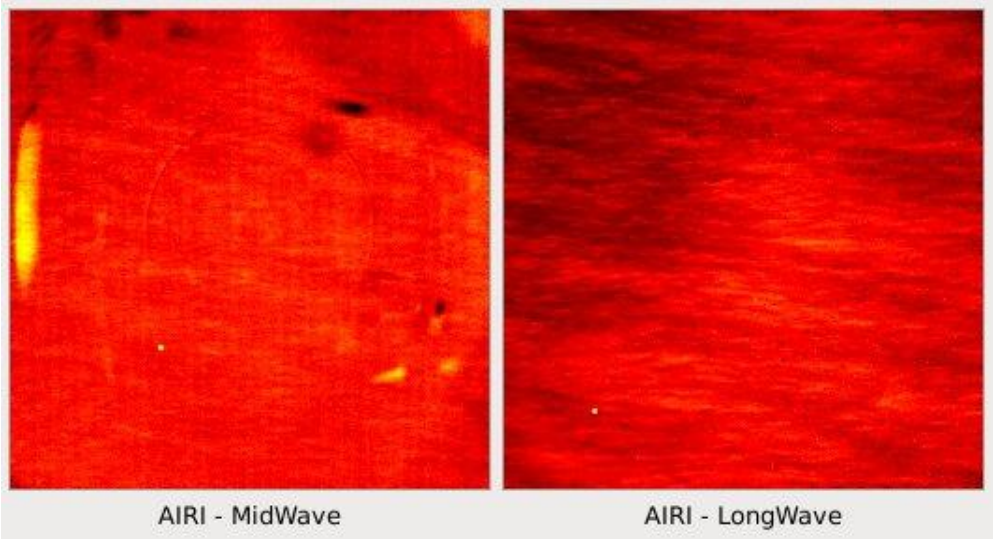

Figure 15: Example low clutter: Imagery from the ocean data set of Flight 826 displaying an ocean background. This data set is classified as a low clutter scenario. An inserted target can be seen towards the bottom left of both frames.

A medium clutter scenario was categorized as one where there is a small amount of high intensity background that could potentially disrupt the tracker's performance. An example of this scenario is from the runway data set taken from Flight 828 shown in Figure 16. The runway located in the middle of the frame can diminish the performance of the tracker because of its high intensity. An inserted target can be seen towards the bottom left of each frame.

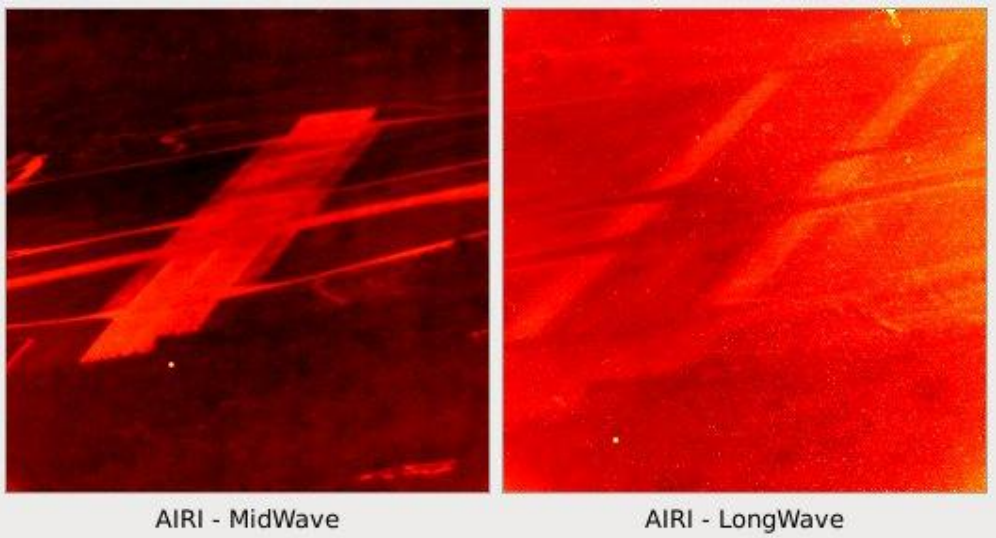

Figure 16: Example medium clutter: Imagery from the runway data set of Flight 828 displaying a forest background with a runway located in the center. This data set is classified as a medium clutter scenario. An inserted target can be seen towards the bottom left of both frames.

A high clutter scenario was categorized as one where there exists a large amount of high intensity background that will most likely impede the tracker's success. An example of this scenario is from the city data set taken from Flight 826 shown in Figure 17. The houses located in the bottom half of the frame provide high intensity points from roofs, roads, and other high temperature objects. The city skyline can be seen in the distance toward the center of the frames and a low intensity river is located towards the top of the frames. An inserted target can be seen toward the top right of each frame.

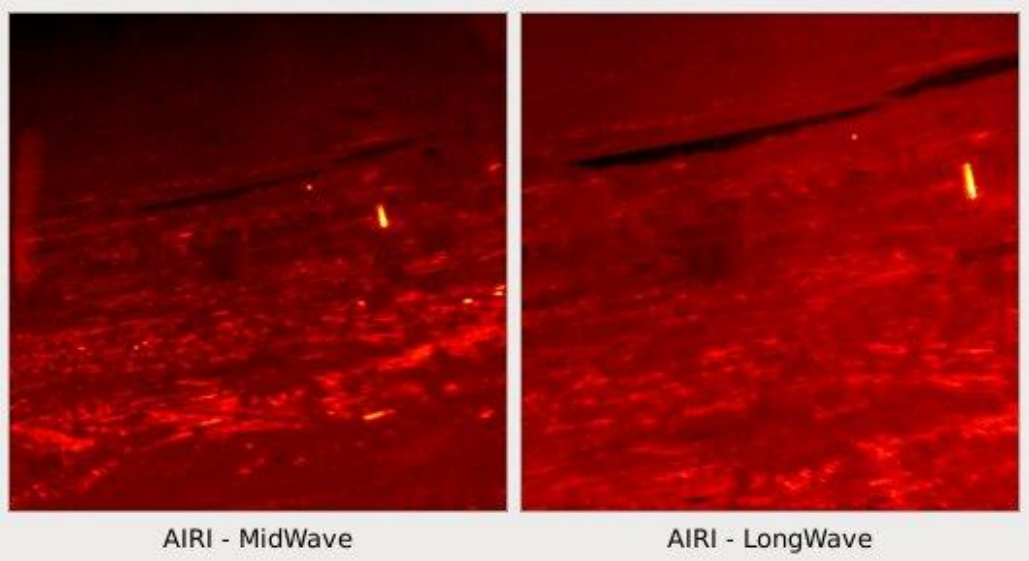

Figure 17: Example high clutter: Imagery from the city data set of Flight 826. The city skyline can be seen in the distance toward the center of the frames and a low intensity river is located towards the top of the frames. An inserted target can be seen toward the top right of each frame.

3.2.2 Inserted Targets

After evaluating the characteristics and behavior of targets within imagery that we viewed containing targets, we determined a systematic method for target insertion. Most of the targets we studied were small cluster targets, that is, targets that appear as a small group of pixels, rather than an aircraft that is resolvable by the human eye. Since the targets are represented as points within a frame, we used a Gaussian low pass filter function in MATLAB to manipulate pixels of a frame. The Gaussian function is represented by the following equation:

$$h_g(n_1, n_2) = e^{-\frac{{n_1}^2+{n_2}^2}{2\sigma^2}}$$

$$h(n_1, n_2) = \frac{h_g(n_1, n_2)}{\sum_{n_1}\sum_{n_2} h_g}$$

In these equations h is the Gaussian function that is created. The n_1 and n_2 represent the x-y indices in h while the σ is the standard deviation of the function (in pixels). The Gaussian function is shown in Figure 18.

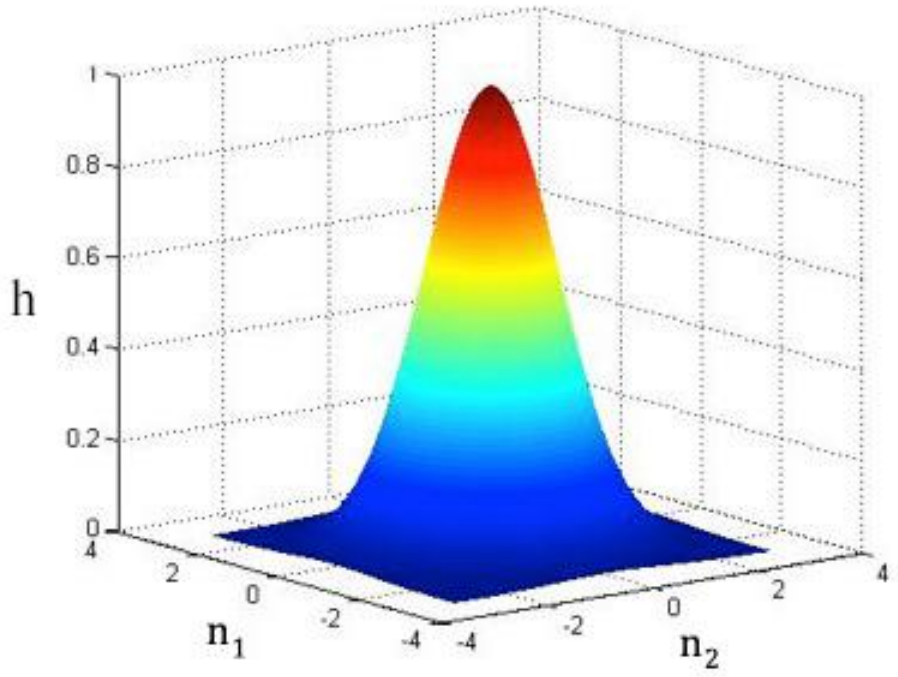

Figure 18: Target insertion Gaussian plot. n_1 and n_2 are the pixel coordinates while the z axis represents the intensity. This example shows a 5 x 5 Gaussian Function with a $\sigma = 0.5$.

The target was inserted by adding the product of the input signal and Gaussian filter to the existing pixel values in a certain range. This is shown in the following equation

$$z_{new}(x, y) = z_{old}(x, y) + h(n_1, n_2) \cdot s.$$

The $z(x, y)$ represents the intensity value of the pixel at the coordinate (x, y) in the frame. The Gaussian is represented as $h(n_1, n_2)$ and the input signal is s. We determined that the small cluster targets vary in size, from two pixels to six pixels. We decided to truncate the Gaussian function and chose a baseline of three pixels in diameter ($n_1 = n_2 = 3$) for our inserted targets. We chose an input signal of the target as a constant level of ten standard deviations above the mean of each image to have a consistent target throughout the data sets. Through our observations, we found actual targets ranging from intensity values of 5 to 100 standard deviations above the mean of the image but ultimately chose ten standard deviations to provide a challenging scenario for our tracker.

An example of the imagery we used to characterize targets is shown in Figure 19. These images, taken from Flight 868, show a clearly defined target with no signs of clutter. The target can be seen as the high intensity bright group of pixels in the top right of each frame.

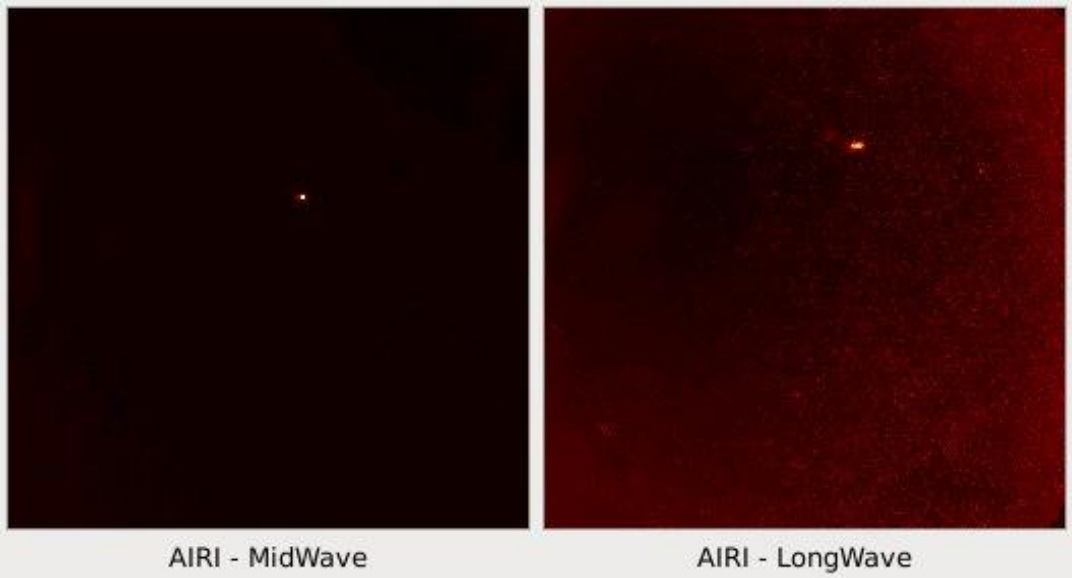

Figure 19: Flight 868 Imagery. An example of blue-sky environment. The actual target can be seen as the group of bright high intensity pixels in the top right of each frame.

From a state file containing information gathered by GPS and INS sensors aboard the aircraft, we were able to determine the change of camera movement in angular space between each frame. The AIRI pointing vector, gathered from sensor information and INS measurements, contains the yaw, pitch, and roll of the sensor measured in radians in the East-North-Up coordinate frame. The yaw of the sensor is denoted by the vertical axis vector out of the top of the sensor and describes which direction, left or right, the sensor is pointing. The pitch is determined from the lateral axis vector pointing out the left side of the sensor. The pitch describes the direction, up or down, that the sensor is pointing. In AIRI's case, the gimbal can only manipulate its yaw and pitch but cannot roll; therefore, the roll output is that of the ASTB platform. The yaw and pitch can be described as angles in radians known as the azimuth and elevation respectively.

To ensure that the target moves realistically between frames, we created a linear target path. The linear path starts in the bottom left of the image and crosses the frame to the top right through the following equations

$$\text{target}_{\text{az}} = (\text{step size})(\text{frame number}) + \text{target}_{\text{az start}},$$

$$\text{target}_{\text{el}} = -(\text{step size})(\text{frame number}) + \text{target}_{\text{el start}}.$$

For a step size, we used $1.6 * 10^{-6}$ radians since the smallest IFOV of either camera was $1.2 * 10^{-4}$ radians which is approximately 75 times larger than our step size, ensuring that

through the 50 frames of data the target would not leave the scene but would pass through most of it. To convert the angular space to a pixel position in our frames, we compared the AIRI pointing vector with the linear target path. First, we set the camera pointing angle to the center of the frame, in azimuth, elevation, and roll, as shown in the following equation

$$\text{AIRIpointing} = (\text{azimuth}, \text{elevation}, \text{roll}).$$

Since the target position on screen is represented by only a relative azimuth and elevation angle, the roll was not considered in the calculations. We then compared the target position to this camera center by taking the difference in radians between the azimuth and elevation in the East-North-Up coordinate system.

$$\text{TargetPosition} = (\text{target}_{\text{az}}, \text{target}_{\text{el}}),$$

$$\text{RelativeTargetPosition}(\text{az}, \text{el}) = \text{TargetPosition} - \text{AIRIpointing}.$$

To convert the angle space to pixel space we used the dimensions of the frames and the IFOV to determine the angular distance spanned by each pixel. In our case, all of our imagery was taken using the AIRI MWIR camera and new LWIR camera, setting the dimensions of the frames to 256 x 256 pixels and the IFOV to $2.0 * 10^{-4}$ radians for the MWIR camera and $1.2 * 10^{-4}$ radians for the new LWIR camera. To calculate the final target positions, we used

$$\frac{\text{RelativeTargetPosition}(\text{az})}{\text{IFOV}} = \text{x}_{\text{pos}},$$

$$\frac{\text{RelativeTargetPosition}(\text{el})}{\text{IFOV}} = \text{y}_{\text{pos}},$$

$$\text{final}_{\text{x}} = \frac{\text{x}_{\text{dimension}}}{2} + \text{x}_{\text{pos}}\,, \text{final}_{\text{y}} = \frac{\text{y}_{\text{dimension}}}{2} + \text{y}_{\text{pos}}.$$

The frames have an origin of (0, 0) at the top left of each frame and increase in value positively both down and to the right. With this information we inserted targets such that they would be in the same relative position between MWIR and LWIR imagery in a scene. By inserting the targets, we were able to export the exact locations, known as truth data, of the center of the target for the analysis of our tracker's performance.

3.3 Image Processing

See Internal Laboratory report. This text not repeated here.

3.4 Target Tracking

The primary goal of the video tracker is to provide analysts in Group 108 with the capability to track a target in IR video during PMA. Identifying the location of the target in each frame allows analysts to extract the target's IR signature.

Figure 20: Target tracking block diagram. The input to the target tracking is an IR image. This image is thresholded and converted to a binary image. The remaining pixels are clustered and the closest centroid is found outputting the target location.

3.4.1 Target Identification

For the target identification step, we implemented an adaptive global thresholding technique, as described in Section 2.4.1, with a threshold level of two standard deviations above the mean of the image. We chose this level as a baseline as this eliminates low intensity clutter without eliminating any possible high intensity targets. For our clustering algorithm we passed a 5 x 5 matrix of "1's" over the pixels to find clusters of nine or more pixels denoted by a "1", as described in Section 2.4.3.

3.4.2 Tracking

The tracking algorithm calculates the distances between the last known track position and all of the clustered points within a rectangular region four times the size of the target box centered at the last known position. The target box is the user initiated box that follows the target frame to frame to show the user the output of the tracker. The target box has a default size of 11 x 11 pixels and can be resized by the user in both the height and width. The last known track position is denoted by the (x, y) pixel coordinates of the target on the previous frame. The clustered points are each "1's", and can be visualized as white pixels in a binary image. The tracker finds the minimum distance between the previous track position and these clusters and records the closest cluster's position. The tracker then finds the centroid of the closest cluster. To find the centroid, the tracker searches a 9 x 9 box around the recorded cluster position. Within the 9 x 9 box, the position of every "1", or white pixel, is recorded as long as the x and y position are not separated from the cluster by a "0." The centroid of the closest cluster is the new track

position. These track positions, as (x, y) coordinates, are saved and can be exported for future analysis. A visual representation of the tracking is shown in Figure 21. For simplicity, the example below is of a 6 x 6 image with a 3 x 3 target and a cluster search range of 5 x 5 pixels.

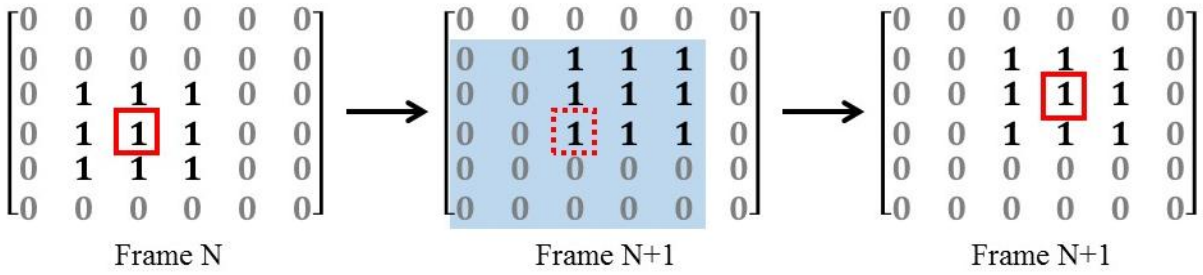

Figure 21: Tracking and centroiding visual representation. Each pixel value is denoted by a "0" or "1" from the thresholding and clustering algorithms. Frame N (left) represents a frame where a target has been centroided, showing the center pixel within the red box. When the frame is switched, denoted by Frame N+1 (middle), the tracker finds the closest cluster position (dotted red box) in a cluster search region (light blue background). The centroiding algorithm then finds the center pixel of the target (red box on right) and outputs its location.

In the frames N and N+1, each pixel is denoted by a "0" or "1" from the thresholding and clustering algorithms. Frame N on the left represents a frame where a target has been centroided, showing the center pixel within the red box. When the frame is switched, denoted by Frame N+1 in the middle, the tracker finds the closest cluster position, denoted by the dotted red box, to the previous track position in a cluster search region, denoted by the light blue background. The centroiding algorithm then finds the center pixel of the target, shown on the right, again denoted by the red box. The location was extracted and exported for analysis.

3.5 Tracker Analysis

In order to analyze the tracker's performance, coordinates of the track positions were exported into a comma-separated value file for later use. The performance was evaluated by calculating the tracker's failure rate and accuracy. We automated the process of collecting data by importing the truth data and running the analysis using a MATLAB script on each data set.

3.5.1 Tracker Failure Rate

The TFR was calculated by finding the number of times track loss occurred in a set of frames. Track loss is defined as the number of times the tracker required manual intervention to correct it. Manual intervention was required when the target intersected or passed outside the target box perimeter. If a target intersected the perimeter of the target box, there was no way to

accurately calculate the IR signature, and therefore it was denoted as lost. The SNR calculation, as described in Section 2.2.1 uses the perimeter of the target box to compute the SNR. Therefore, if the target is not completely within this perimeter, then the calculation would be invalid. The equation for the tracker failure rate was calculated in the following manner:

$$\text{TFR} = \left(\frac{\#\text{ of frames requiring manual intervention}}{\text{total }\#\text{ of frames}}\right) \times 100\%$$

From this metric, we determined top-level performance of our tracker and quantified its effectiveness. The two-dimensional root mean square error (RMSE) was calculated for those tracks which fall inside the track box.

3.5.2 Track RMSE

The track error is the two-dimensional position error in pixel (x, y) space that was calculated by comparing the tracker outputs to the truth data. These pixel errors were presented as (x, y) signed errors. The RMSE of the track error is given by

$$\text{RMSE} = \sqrt{\frac{1}{n}(\Delta x_1^2 + \Delta x_2^2 + \Delta y_1^2 + \Delta y_2^2 \cdots + \Delta x_n^2 + \Delta y_n^2)},$$

or more simply in terms of distance

$$\text{RMSE} = \sqrt{\frac{1}{n}(\Delta d_1^2 + \Delta d_2^2 + \cdots + \Delta d_n^2)}.$$

The RMSE of the track error in both x and y directions represents the dispersion of track error. The norm of the RMSE in each direction is taken to consolidate the two accuracy metrics into a single value to be reported for each combination of background clutter scenario and processing technique. The RMSE is zero when all tracks for a sequence of frames lie at truth.

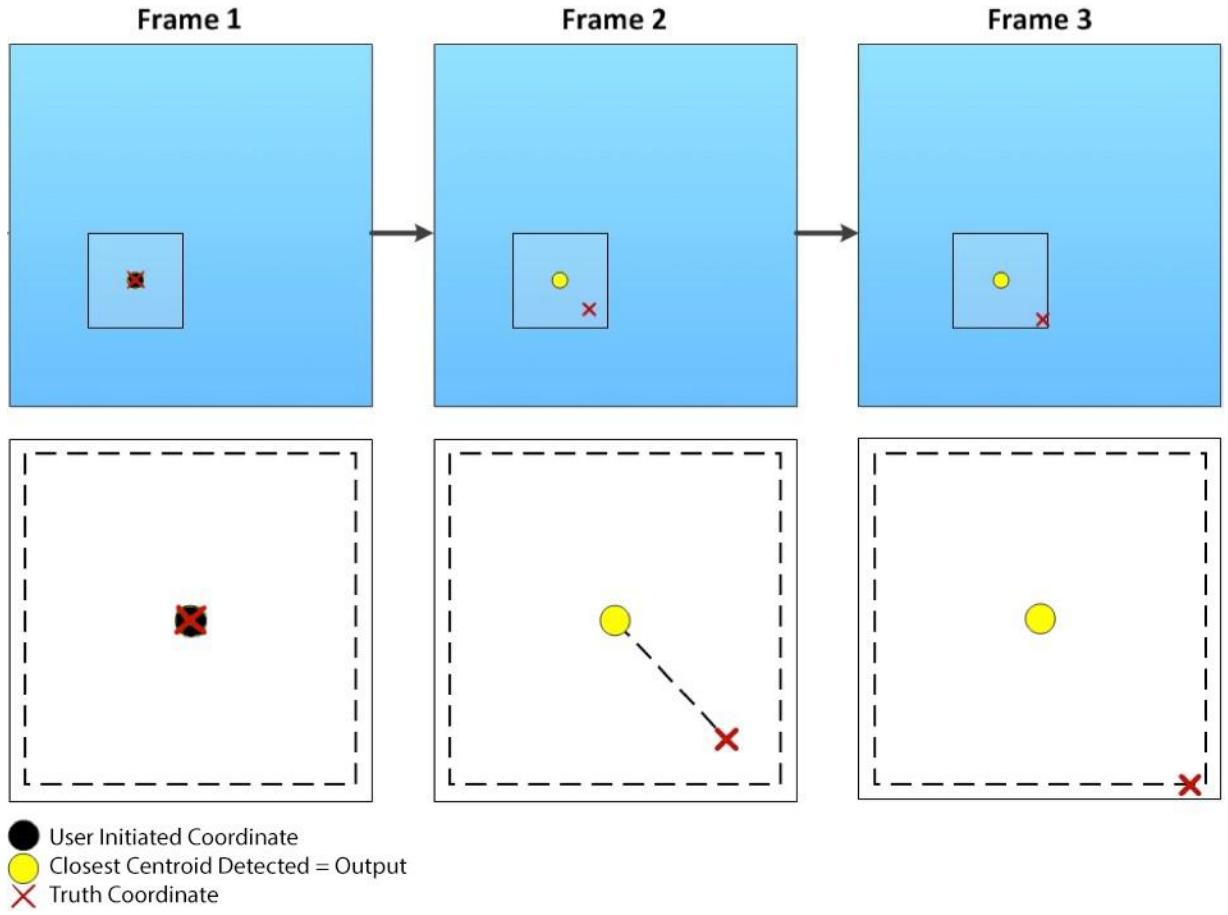

Figure 22: Tracker accuracy description. Frame 1 denotes the user initiated first frame. Frame 2 denotes any frame where the target has been successfully tracked. Frame 3 denotes a frame where the tracker has lost the track of the target.

In Frame 1 of Figure 22, the user initializes the target tracker. When a user clicks on the screen, a target box appears around the coordinates pressed, denoted by the black circle. Frame 2 shows that the tracker output a location (yellow dot) near the truth (red x). The tracker used the centroiding algorithm to center the target box on the center of the closest high intensity cluster, represented by the yellow circle, and output its x and y coordinates for the accuracy calculation. The dotted black line represents the error between the tracker output and truth value. Frame 3 is a representation of a frame where the track of the target has been lost. If the truth target is partially or fully outside of the target box, it was no longer considered tracked; therefore the target track was denoted as lost and no track error was recorded for this frame.

4 Results

See internal Laboratory report. This text not repeated here.

5 Discussion

See internal Laboratory report. This text not repeated here.

6 Conclusions

See internal Laboratory report. This text not repeated here.

7 References:

Banerjee, S. (2008, January 20). How to compute a homography. Retrieved October 2, 2014, from http://www.cse.iitd.ernet.in/~suban/vision/geometry/node24.html.

Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media, Inc. 163-173.

Common Interfaces of Descriptor Matchers. (2014, April 21). Retrieved October 3, 2014, from http://docs.opencv.org/modules/features2d/doc/common_interfaces_of_descriptor_matchers.html?highlight=bruteforcematcher#bruteforcematcher

Geometric Image Transformations. (2014, April 21). Retrieved October 2, 2014, from http://docs.opencv.org/modules/imgproc/doc/geometric_transformations.html#warpperspective.

Gonzalez, R. C., Woods, R. E., & Eddins, S. L. (2004). *Digital Image Processing Using MATLAB.* Upper Saddle River, New Jersey, USA: Prentice Hall.

Gunner, J. (2010). Falcon 20 ECM of FEKS 717 Skv. Royal Norwegian AF. [Photograph]. Retrieved September 15, 2014, from: http://upload.wikimedia.org/wikipedia/commons/0/04/053_AMD_Falcon_20_ECM_of_FEKS_717_Skv._Royal_Norwegian_AF.jpg

Klick, D., Blumenau, P.M., Grimm, J.G., Knight, F.K., Marker, C.R., Theriault, J.R. Jr. (1996). Airborne Infra-Red Imager (AIRI): A Dual-Band IR Sensor for the Airborne Seeker Test Bed. Massachusetts Institute of Technology: Lincoln Laboratory.

Klick, D. I., Blumenau, P. M., & Theriault Jr, J. R. (2001, September). Detection of targets in infrared clutter. In Aerospace/Defense Sensing, Simulation, and Controls. 120-133. International Society for Optics and Photonics.

Lehar, S. (n.d.) An Intuitive Explanation of Fourier Theory. Retrieved October 2, 2014, from http://cns-alumni.bu.edu/~slehar/fourier/fourier.html.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2), 91-110.

Mahulikar, S. P., Sonawane, H. R., Rao, G. A. (2007, October). Infrared signature studies of aerospace vehicles. Progress in Aerospace Sciences. 43(7-8), 218-245.

Nixon, M. S., & Aguado, A. S. (2008). *Feature Extraction and Image processing.* Oxford: Academic Press.

Richards, M. A. (2014). *Fundamentals of Radar Signal Processing.* New York: McGraw-Hill.

Sinha, U. (2010). SIFT: Generating a feature - AI Shack. Retrieved September 22, 2014.

Szeliski, R. (2011). Feature detection and matching. In R. Szeliski, *Computer Vision: Algorithms and Applications, Texts in Computer Science* (pp. 181-234). Springer London.

Truitt, J. (2006). AIRI NUC Stability. Massachusetts Institute of Technology: Lincoln Laboratory.

Truitt, J. (2007). Varying source size calibration of AIRI. Massachusetts Institute of Technology: Lincoln Laboratory.

Wagner, T. (2013). Advanced Tracker: Summer Intern Project Report. Massachusetts Institute of Technology: Lincoln Laboratory.

Young, H. D., & Freedman, R. A. (2008). Photons, electrons, and atoms. University physics with modern physics (12th Edition ed., pp. 1307). San Francisco, California: Pearson Education Inc.

Appendix A: Data Set Images

This appendix contains one frame of both the MWIR and LWIR imagery for each cluttered data set that have been used in the analysis. Descriptions and frame numbers of each of the data sets can be found in Table 2.

Imagery from the coastline data set of Flight 826 is shown in Figure 23. The bottom of the frames contain low intensity coast while the top of the frames show high intensity ocean and higher intensity glints from the sun's reflection off of the surface. This data set was classified as a medium clutter scenario. An inserted target is visible towards the bottom left of both frames.

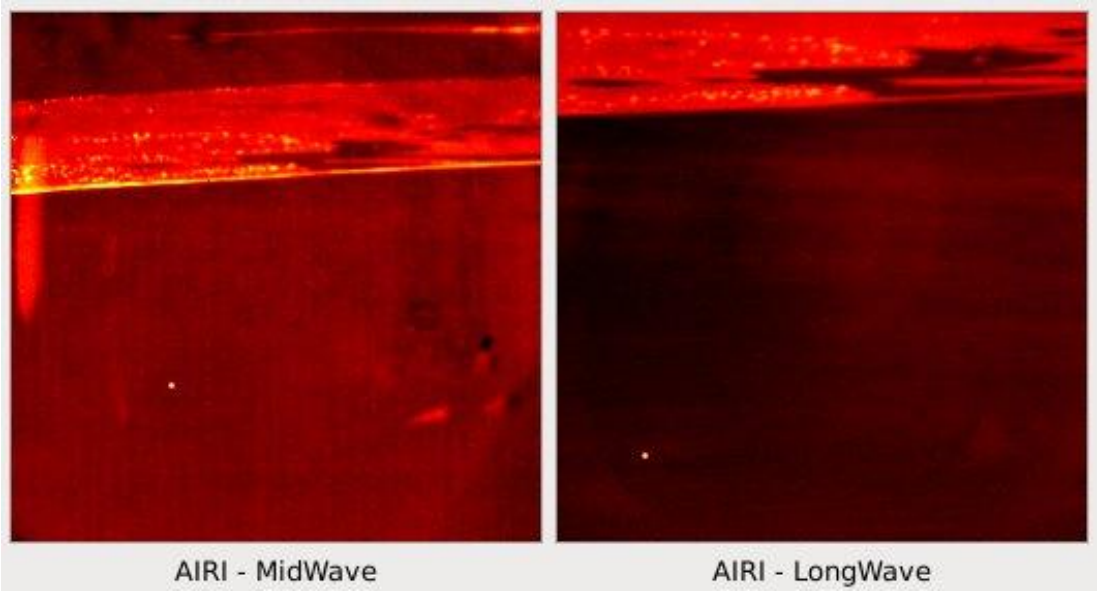

Figure 23: Imagery from the coastline data set of Flight 826.

Imagery from the forest 1 data set of Flight 826 is shown in Figure 24. The frames contain a low intensity arborous background with high intensity artifacts spattered throughout. This data set was classified as a medium clutter scenario. An inserted target is visible towards the bottom left of both frames.

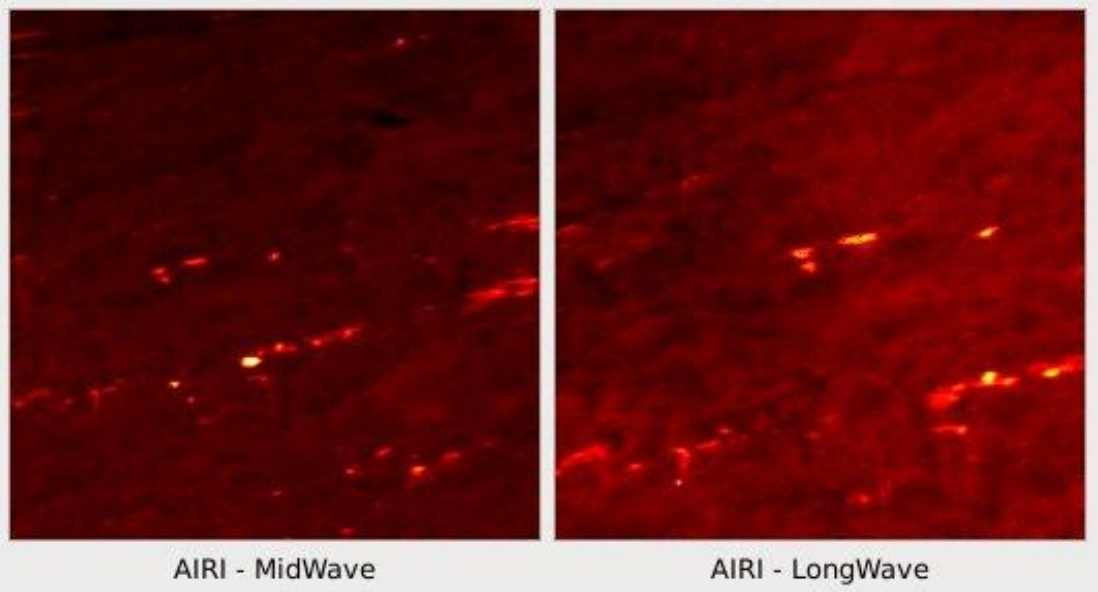

Figure 24: Imagery from the forest 1 data set of Flight 826.

Imagery from the farmland data set of Flight 826 is shown in Figure 25. The frames contain low intensity trees surrounding areas of high intensity farmlands. Buildings are also located towards the top of the MWIR imagery and can be seen throughout the data set. This data set was classified as a high clutter scenario. An inserted target is visible towards the bottom left of both frames.

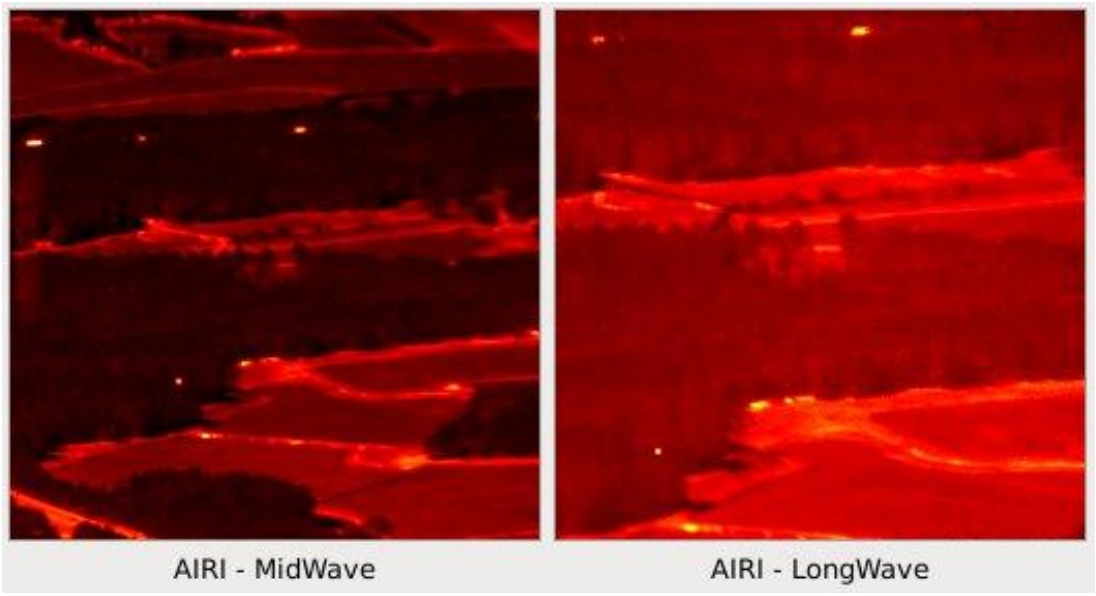

Figure 25: Imagery from the farmlands data set of Flight 828.

Imagery from the ocean data set of Flight 826 is shown in Figure 26. The frames contain a low intensity ocean background. Periodic waves can be seen clearly in the LWIR imagery. This data set was classified as a low clutter scenario due to the low variation in the background. An inserted target is visible towards the bottom left of both frames.

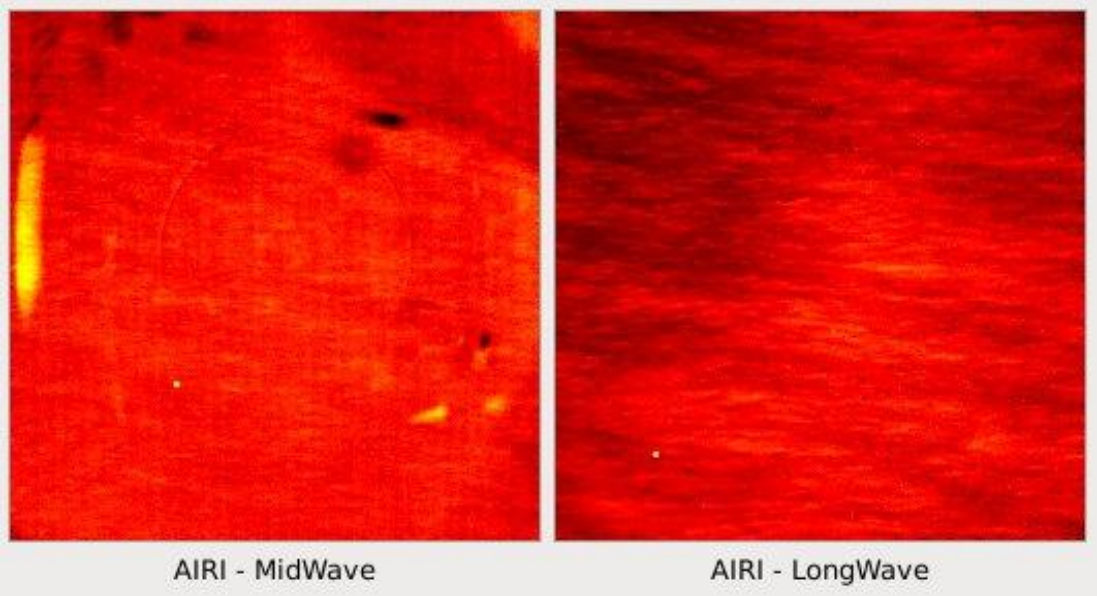

Figure 26: Imagery from the ocean data set of Flight 826.

Imagery from the city data set of Flight 826 is shown in Figure 27. The houses located in the bottom half of the frame provide high intensity points from roofs, roads, and other high temperature objects. The city skyline is visible in the distance toward the center of the frames and a low intensity river is located towards the top of the frames. An inserted target is present toward the top right of each frame. This data set was classified as a high clutter scenario.

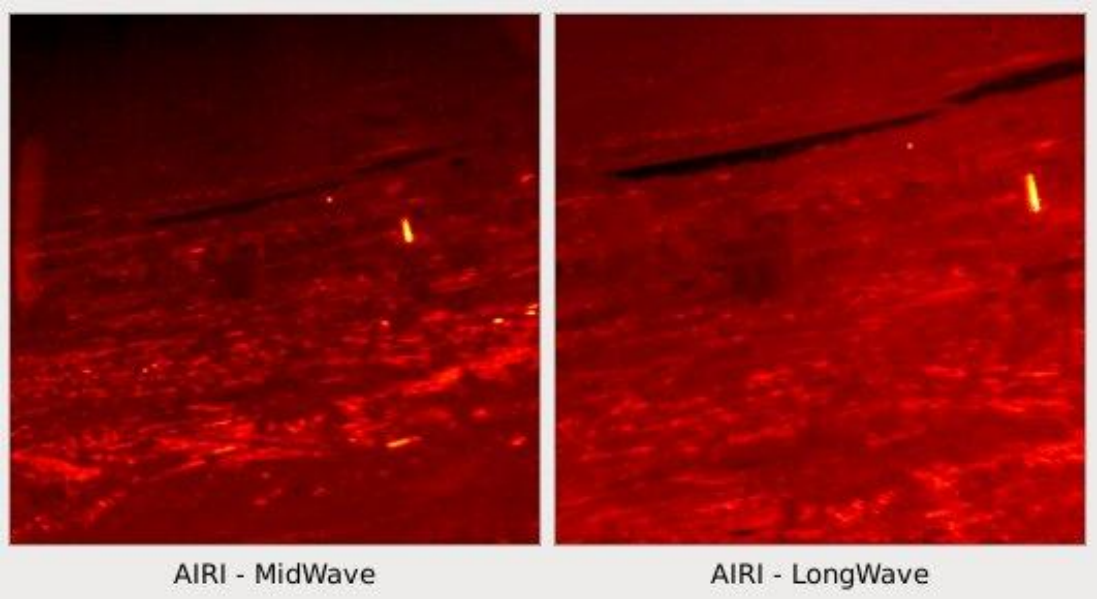

Figure 27: Imagery from the city data set of Flight 826.

Imagery from the forest 2 data set of Flight 828 is shown in Figure 28. The frames contain a low intensity arborous background. This data set was classified as a low clutter scenario due to the low variation in the background. An inserted target is visible towards the top right of both frames.

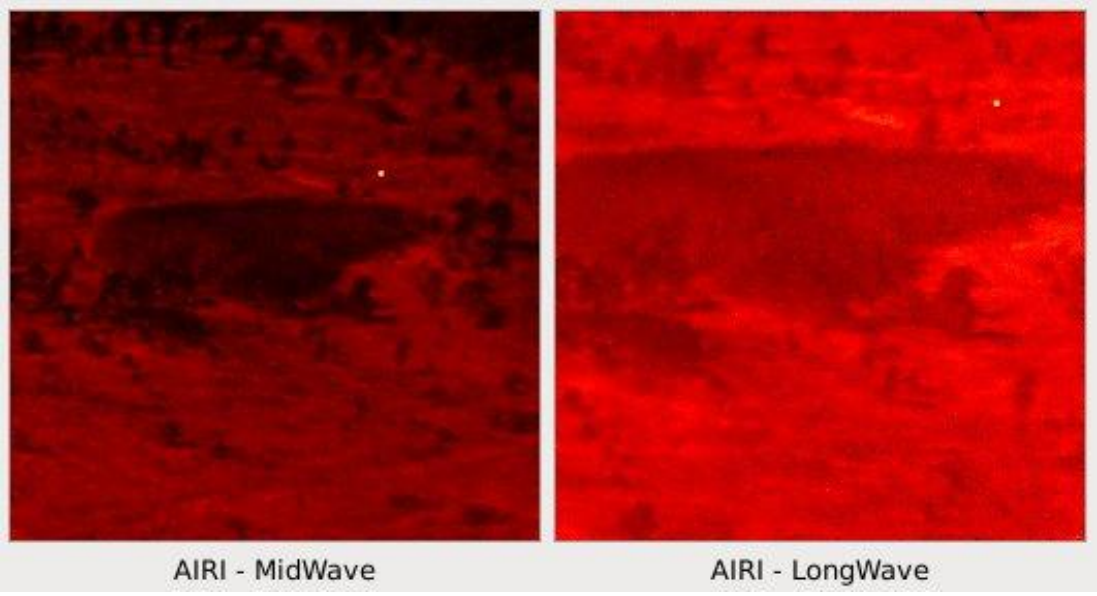

Figure 28: Imagery from the forest 2 data set of Flight 828.

Imagery from the runway data set of Flight 828 is shown in Figure 29. The frames contain a low intensity forest background surrounding a high intensity runway in the center of the frames. This

data set was classified as a medium clutter scenario. An inserted target is visible towards the bottom left of each frame.

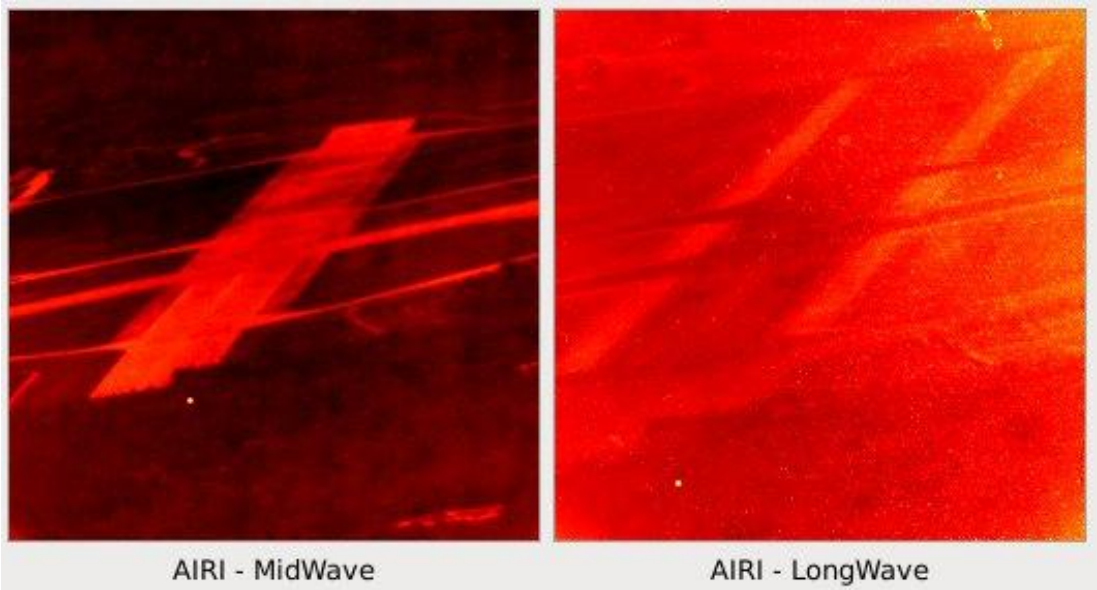

Figure 29: Imagery from the runway data set of Flight 828.

Imagery from the suburbs data set of Flight 828 is shown in Figure 30. The frames contain high intensity houses and roads surrounded by a low intensity forest background. This data set was classified as a high clutter scenario. An inserted target is visible towards the top left of each frame.

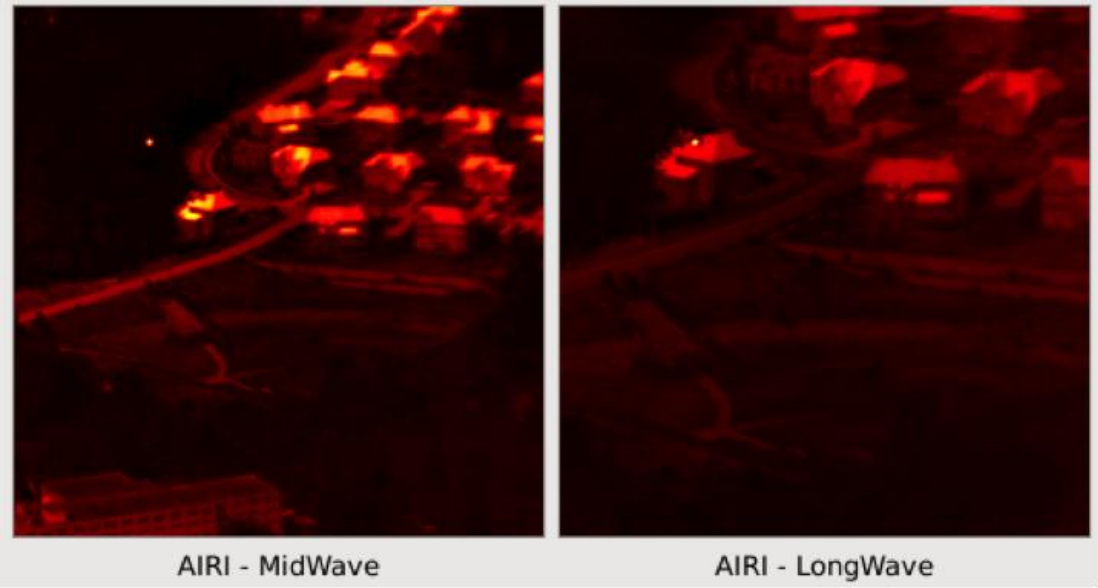

Figure 30: Imagery from the suburbs data set of Flight 828.

Imagery from the airport data set of Flight 838 is shown in Figure 31. The frames contain high intensity buildings, roads, and runways with low intensity grassy areas in between. This data set was classified as a high clutter scenario. An inserted target is visible towards the bottom left of each frame.

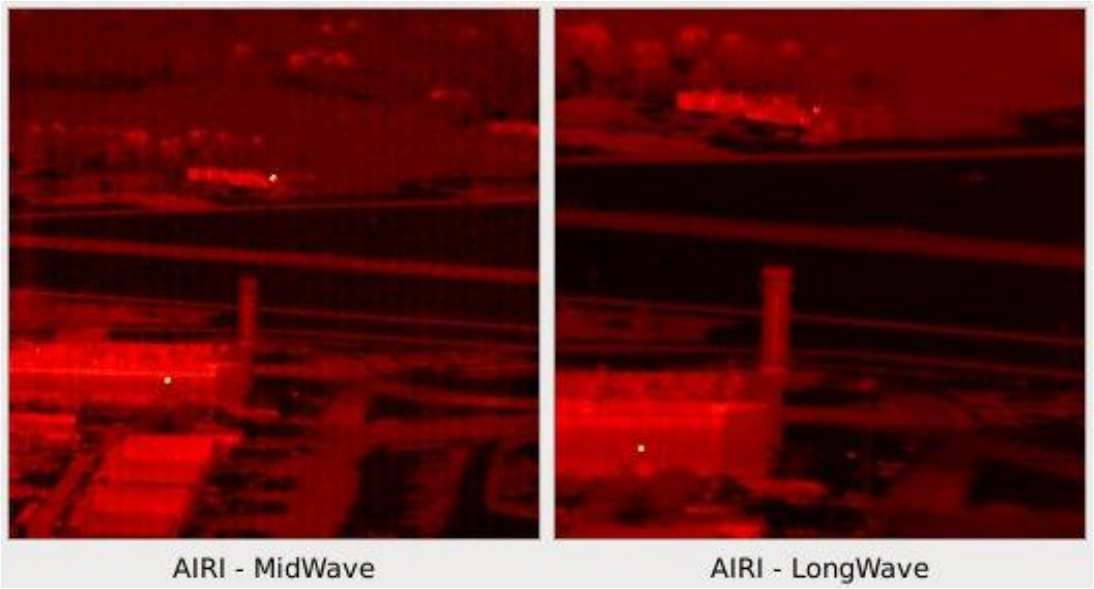

Figure 31: Imagery from the airport data set of Flight 838.

Imagery from the segmented data set of Flight 845 is shown in Figure 32. The frames contain moderate intensity ocean and coastline backgrounds towards the bottom of the frames. This coastline transitions to a city skyline approximately halfway up the frames and then cloud background towards the top of the frames. This data set was classified as a medium clutter scenario. An inserted target is visible towards the bottom left of each frame.

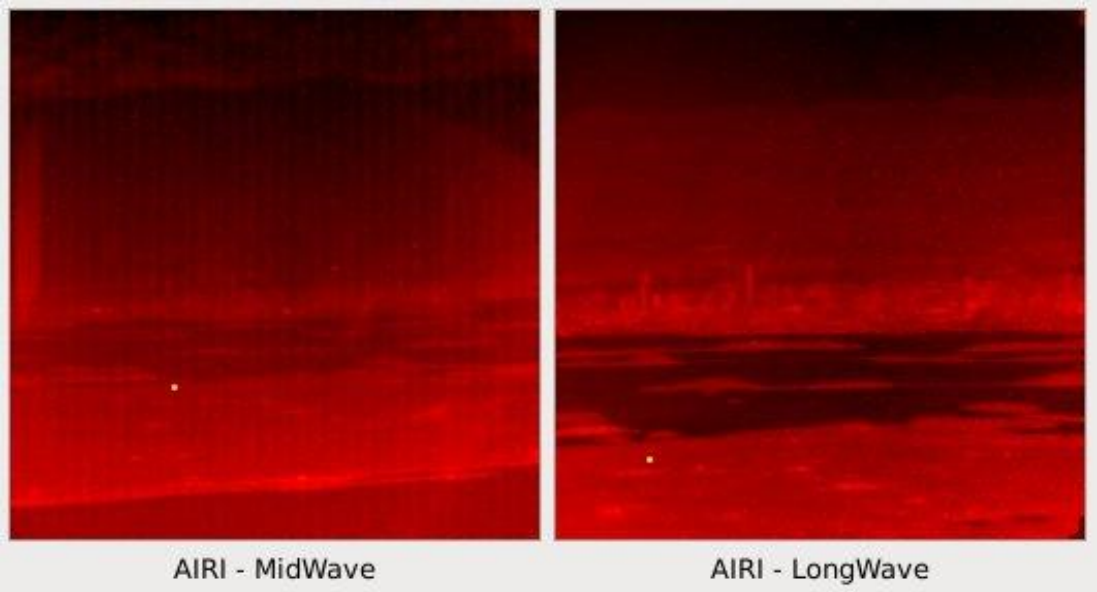

Figure 32: Imagery from the segmented data set of Flight 845.

Imagery from the clouds data set of Flight 868 is shown in Figure 33. The frames contain high intensity clouds with low intensity blue-sky background. This data set was classified as a high clutter scenario. An inserted target is visible in the center of each frame.

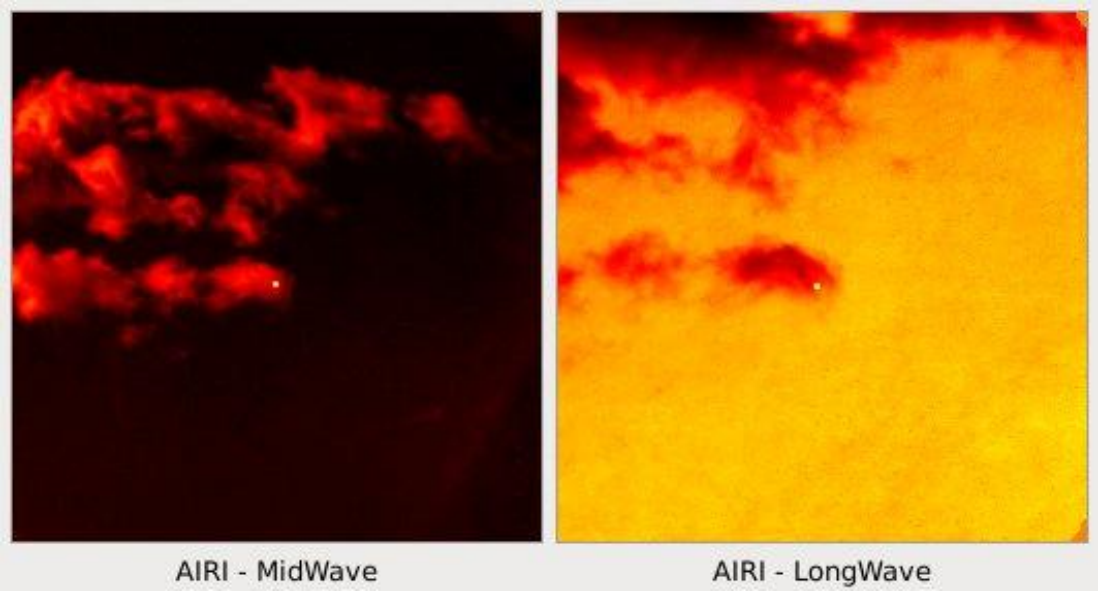

Figure 33: Imagery from the clouds data set of Flight 868.

Imagery from the flat clouds 1 data set of Flight 886 is shown in Figure 34. The frames contain a low intensity cloud-bed. This data set was classified as a low clutter scenario. An inserted target is visible towards the top right of each frame.

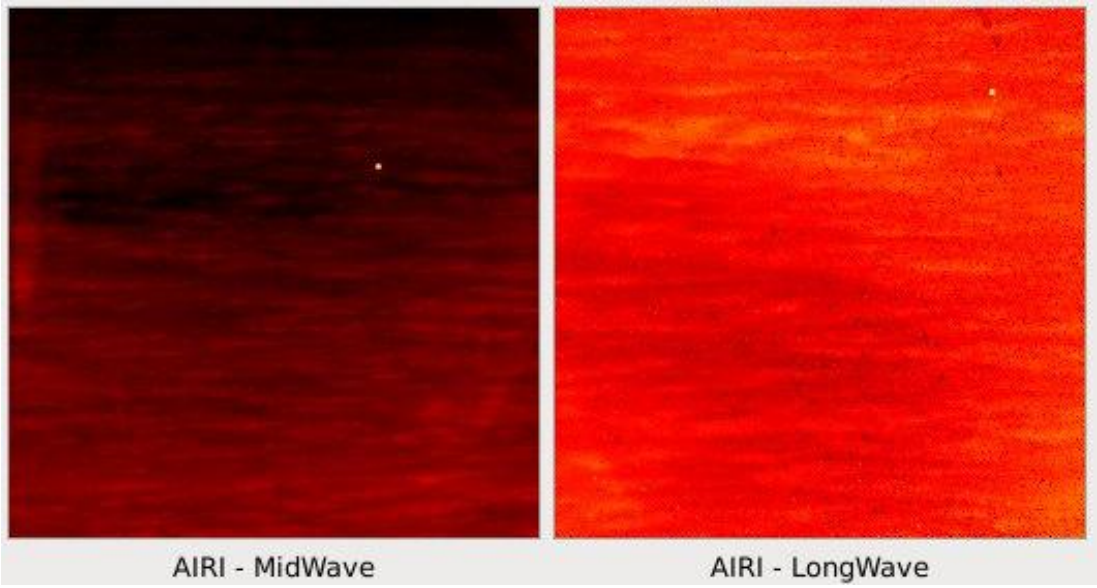

Figure 34: Imagery from the flat clouds 1 data set of Flight 886.

Imagery from the flat clouds 2 data set of Flight 886 is shown in Figure 35. The frames contain a low intensity cloud-bed. This data set was classified as a low clutter scenario. An inserted target is visible towards the top right of each frame.

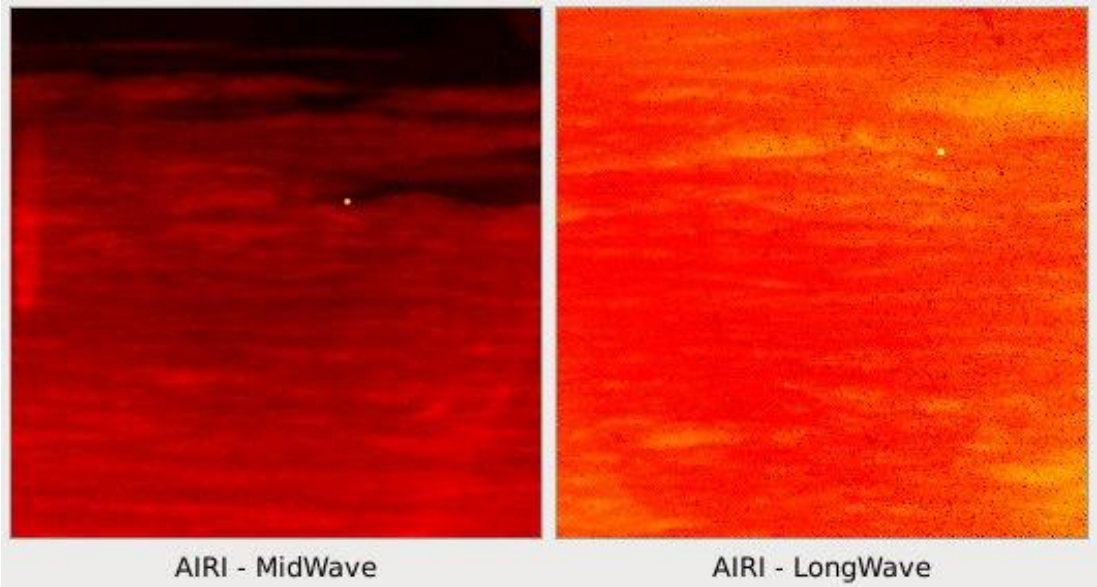

Figure 35: Imagery from the flat clouds 2 data set of Flight 886.

Appendix B: Video Analysis Tool (VAT) GUI Implementation

See internal Laboratory report. This text not repeated here.

Appendix C: Additional Processing Images

See internal Laboratory report. This text not repeated here.